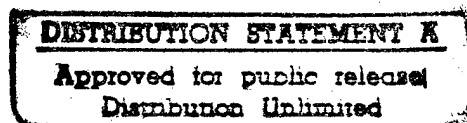


Automated Size Prediction
for
Try-on of U.S. Army Men's Initial Issue Dress Uniform

Final Report

A Short Term Research and Development Task
Proposed Under DLA900-87-D-0017
Delivery Order #0026

June 1992-June 1994



19960726 072

Principal Investigator: Dr. Nancy J. Staples
Clemson Apparel Research
(803) 646-8454

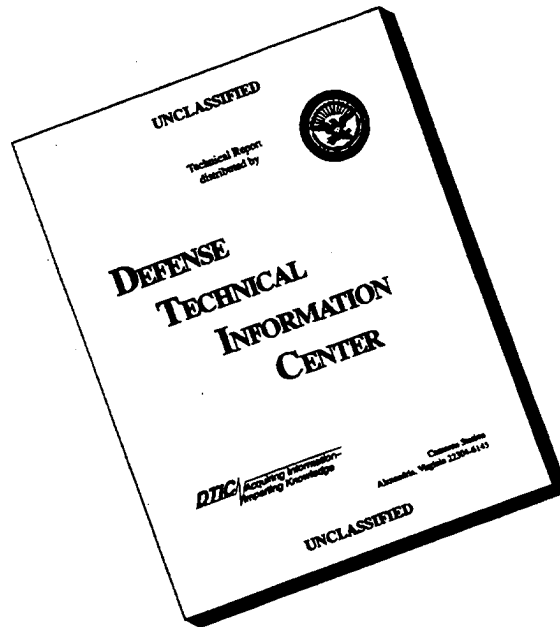
Investigators: Dr. J. Steve Davis
Department of Management
(803) 656-3768

Dr. Roy Pargas
Department of Computer Science
(803) 656-5855

Contractor: Clemson University
Clemson Apparel Research
500 Lebanon Road
Pendleton, SC 29670

DTIC QUALITY INSPECTED 1

DISCLAIMER NOTICE



**THIS DOCUMENT IS BEST
QUALITY AVAILABLE. THE
COPY FURNISHED TO DTIC
CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO
NOT REPRODUCE LEGIBLY.**

Table of Contents

Background	1
Objective	1
Research Plan	2
Preparations	2
Initial Scanning Equipment Availability	5
Size Prediction System Development	5
An explanation of case-based reasoning	7
Preliminary system plans	8
Coding scheme used for recording cases	8
Progress of data entry	10
Checking for validity of cases	11
Determining how to handle cases in ReMind®	11
Run time for predictions	12
Progress of expert system data entry	12
The first prototype prediction system	12
Demonstration at Bobbin Show	13
Improved prototype prediction system	13
System predictions	14
Improved prediction scheme	16
Improved data import	18
Arrangements for continued use of the case-based reasoning tool	20
Review and project planning	20
A field-test-worthy prototype expert system	21
First operational test	23

Table of Contents

Related issues	27
Continued expert system performance evaluation	27
Second operational test	28
Size prediction system learning curve studies	30
First learning curve study	30
Second learning curve study	31
Selection of test cases	32
Forming databases	32
Converting to CBR library	32
Testing	33
Refining the set of test cases	33
Results of second learning curve study	34
Third learning curve study	34
3D body scanner interface software development	35
Point-and-click measurement, circumferences, and straight-line distances	36
Porting code from SUN workstations to IBM PC-compatibles	38
The user interface	38
Point and click surface measurement	39
Single source, multiple destination (radial) measurement	40
Multiple point measurement along the body contour	41
Development of PC software	41
Continued tool development—vertical slices	43
Automatic body part recognition	46

Table of Contents

Tracking 3D body scanning technology	49
Fact finding visit to Wright Patterson Air Force Base	52
Investigation of British scanning devices	52
Automation of armed forces measurement blank	53
Presentations, demonstrations, and related activities	56
Accomplishments	59
Tables	
1. Structure for database: A:\IMPORT.DBF	19
2. Initial Weightings	24
3. Accuracy percentages from test at Fort Jackson, November 10, 1993 for 70 soldiers (overall garment)	26
4. Results of Testing with the Stacked Approach	34
List of Figures	
1. Example of body scan output	6
2. Highlighted slice for circumference measurement	37
3. Highlighted surface distance to be measured (before smoothing)	42
4. Vertical slice selected	44
5. Vertical slice displayed in side view	45
Appendices	
A: Student thesis, Vemuri	
B: Student thesis, Jindal	
C: 3DM documentation	
D: 3DM source code	
E: Student thesis, Knight	

Table of Contents

Appendices

F: Student paper, Sen

G: Technical paper submitted, OLE

H: Technical paper submitted, IEEE

I: Newspaper article, Greenville News, June 3, 1993

J: Published trade article, AIM August 1994

K: Published trade article, AIM October 1994

L: Size prediction CDRLs

M: Measurement extraction CDRLs

Background

The current manual system for selecting uniform sizes for try on is inefficient due to errors in measurement of soldiers and repeated trials of various sizes of garments. Inaccuracies in measurement generally result from the improper placement of the measuring tape and from variations in its tension. This is exacerbated by the use of soldiers on detail assigned the task of measuring. Even among experienced fitters, measurements are inconsistent—measurements taken by the same fitter may vary in consistency when the fitter gets tired. Although size prediction charts are available, they are not effectively used because of the fitters' lack of confidence in the accuracy of the measurements and the difficulties of accounting for priorities among body dimensions. Instead of relying on the prediction charts, a "you look like this size" approach is generally employed.

In preliminary tests at Fort Jackson, SC the feasibility of employing an expert system to determine garment sizes was demonstrated through the use of a student project, rule-based expert system for pants. The prototype expert system was quick and easy to use, and, in spite of detail soldiers taking the measurements, it selected the correct garment size for 50% of the 100 soldiers involved in the test. An expert system has the capability of prioritizing measurements to select the correct or most easily alterable size and it never gets tired.

The tests also indicated a potential for significant time saving. Based on data collected at Fort Jackson, a correct selection of trouser size by the expert system could reduce fitting time to less than half the average time required by the current manual method. Saving time not only reduces total processing time, but also achieves significant cost savings in Clothing Initial Issue Point (CIIP) operations. Even if an expert system selected the correct size only 60% of the time, the reduction in time to fit 20,000 soldiers for trousers alone would be 100 hours per CIIP employee, or \$1000 each based on an average of \$10/hour. These savings could be realized at a single installation during the first year.

Incorporation of a 3-dimensional, non-contact measurement device could further improve the fitting process. The selection accuracy of the expert system would be significantly improved if it were provided accurate, consistent measurements. The need for better accuracy was reinforced by the finding at Fort Jackson that, for a random sample of soldiers included in the study who were re-measured by trained fitters, 80% had been inaccurately measured by detail soldiers.

Objective

The objective of this study was to automate the prediction of US Army male dress uniform initial issue try-on size by employing an expert system in coordination with accurate 3-dimensional, non-contact body measurement.

Research Plan

The CIIP at Fort Jackson, led by Mr. Lonnie Turner, Chief, has been most cooperative and agreed to participate with Clemson in the development and testing of any procedure that would expedite the clothing initial issue process. The Fort Jackson CIIP currently processes approximately 200 soldiers per day with an average alteration cost of \$25 per soldier.

The research plan was in Phase 1 to develop and calibrate expert system software which uses body dimensions as input and predicts the appropriate garment size for try on. With the cooperation of Mr. Turner's staff, the system would be tested under conditions in which measurements have been taken by fitters, not detail soldiers. Meanwhile, interactive software to make the output of a 3-dimensional, non-contact body scan useful for extracting body measurements would be developed. When the expert system was running smoothly, and the measurement extraction software developed, Phase 2 would include the incorporation of a 3-dimensional, non-contact measuring device to replace manual measuring. Researchers would work interactively through a computer interface with the measuring equipment to select the locations where body dimensions should be measured. These data would be electronically fed to the expert system and the size predictions would be printed out for each soldier. The time required for these operations and the level of prediction accuracy would be recorded and analyzed.

Preparations

The project to design an expert system for initial try-on of the US Army men's dress uniform began on June 10, 1992 with CAR being notified that the contract had been awarded. That same day Dr. Nancy Staples and Dr. Roy Pargas called Peter Kuhlman, President of Dimensional Measurement Systems, to begin making arrangements for using their 3-D, non-contact body measurement equipment. Dr. Staples also contacted Jack deRaines at Gerber Garment Technologies to set up a meeting date (July 7) to co-ordinate our respective DLA projects to avoid overlap and to determine needs, if any, for linkages between the two projects to be developed.

Dr. Steve Davis and graduate students Sarat Vemuri and Murali Earagolla began an investigation of the latest methods and software tools for developing expert systems to determine which one would be the most suitable for this project. The newly-released case-based-reasoning (CBR) approach (developed through a contract with DARPA in 1987) appeared particularly appropriate. The use of this technology in the automated size prediction project was one of its first applications.

Dr. Roy Pargas initiated a review of the state-of-the-art in 3-D measuring devices to compare with the DMS equipment being considered for the project.

He contacted and received information from the following: Robotic Vision Systems, Inc., Industrial Perception Systems, Technical Arts, Technological Artisans, and Jandel Scientific.

On June 15 and 16, at the Naval Training Center in Orlando, Florida, Dr. Staples briefed the joint working group of the customer-driven uniform manufacturing project to solicit their suggestions and their support for the automated size prediction project. While at NTC, she observed the Navy method of assigning uniforms.

During the week of June 22, plans were made for a trip to Fort Jackson, SC on June 29, 30, and July 1. The purposes of the trip were to familiarize Davis, Vemuri, and Earagolla with the Fort Jackson Clothing Initial Issue Point (CIIP), its personnel and procedures, to update Mr. Lonnie Turner, Chief, CIIP, and his staff on the status and plans for the project, to collect body measurement and sizes assigned data on soldiers processed on those three days, and to collect body dimensions, sizes assigned, and garment dimensions for a random sample of soldiers processed on two days. A dBase file was created and loaded on a notebook portable computer for the recording of soldier information. Arrangements were made for Dr. Staples to be allowed to take male soldiers' body measurements.

On June 29 and 30, the team observed all phases of the Fort Jackson CIIP operation, toured the warehouse, and recorded selected portions of the operation on video for later analysis. The team collected and input on the computer 1) body measurements, sizes assigned, garment dimensions, and number of try-ons for 54 randomly selected soldiers and 2) body measurements with accompanying sizes assigned for a total of 426 soldiers.

The random sample data were collected in order to determine the relationships between body dimensions and actual sizes selected, as opposed to sizes predicted by the current U. S. Army size prediction charts. The number of try-ons were recorded to help establish a baseline for the efficiency of the current system. The actual garment dimensions were checked against current specifications to determine if they were within tolerance and will be used as an indicator of the preferred difference between body and garment dimensions for comfortable fit while meeting regulations for appearance. The larger body of data on all soldiers processed were used as case data in the development of the Phase 1 system to help predict garment sizes.

The Fort Jackson trip culminated with the collection of records of 222 soldiers on July 1. During the early part of the day on July 1, Dr. Nancy Staples, Dr. Steve Davis, and graduate students Sarat Vemuri and Murali Earagolla visited the Parris Island US Marine facility to observe their clothing initial issue process and discuss possible co-ordination with the Marines on their potential use of automated size prediction. Mr. Dave Meadley briefed the team on his vision for automating the Marine clothing issue process and Captain Tom Saldana led the team on a tour. Because the Marines do not record body dimensions (the fitter measures the body and tells the recorder a

size for try-on, not the body dimension), the team concluded that the Marines would have a greater need for the output of this project after the computerized body measuring was incorporated. Mr. Meadley and Captain Saldana are being briefed on the progress of the project and want to observe the field test of the expert system with manual measurement-taking.

On July 8, Dr. Staples met with Jack deRaimis, Bob Tuttle, and Paul Clarke at the South Windsor, CT office of Gerber Garment Technologies to be briefed about the content of the GGT subcontract with EFFI at FIT sponsored by DLA. Dr. Staples then briefed the GGT people about the size prediction contract. It was determined that the two projects do not overlap, but that by summer 1993 there should be opportunities to use a portion of the output of the size prediction project with the GGT lower torso prototype expected to be developed by then.

On July 9, Dr. Staples visited Natick RD&E labs. Discussions were held with Steve Israelian, Dan deLuis, Tony Pingiaro, and Barbara Quinn about the size prediction project. Their assistance was solicited in beginning to consider the kinds of potentially useful, never-before-accessible data that can be provided by 3-D, non-contact body measuring equipment. For example, it will be possible to create a horizontal section across the broadest part of the shoulders and upper arms such that an oval will be defined, the long and short axes of which may help determine the need for a larger size jacket than the chest measurement alone would dictate. Plans were made for Dan deLuis to make trips to CAR and to Fort Jackson to observe and participate in field tests of the output of each phase of the project. It was suggested that, during the field test of the first phase, a project team member be stationed at the end of the issue line to measure the garment dimensions of all those garments issued in a size other than that predicted, in an attempt to determine the reason for that particular size being selected.

While at Natick, Dr. Staples was introduced to and had an opportunity to speak with Dr. Steve Paquette, an anthropologist using 3-D modeling for human factors and ergonomics research. Dr. Paquette was briefed on the size prediction project and its application of 3-D, non-contact body measurement. The differences between body dimensions needed for ergonomics and human factors applications versus garment pattern development applications were discussed.

On July 10, Dr. Staples visited Mr. Eugene Zarzycki at the cadet uniform factory at West Point to observe their implementation of an Investronica CAD system and to discuss potential collaboration in additional field tests of the output of the size prediction project. Dr. Staples also had the opportunity to meet Captain Bill Barrige who was in his first week as Chief of Cadet Services replacing Major Kent Lester, who had just transferred to another position.

Initial Scanning Equipment Availability

On July 16, 1992 Mr. Ed Hill and Drs. Staples, Davis, Pargas, and Peck from CAR and Mr. Don O'Brien of DLA met in Charlotte with Mr. Joe Off and Mr. Jud Early of [TC]² to discuss CAR's use of the Dimensional Measurement Systems, Inc. 3-D, non-contact body measuring equipment in which the Department of Commerce, through [TC]² had invested R&D funds. It was determined that the size prediction project's need for the equipment would not conflict with that of [TC]²'s. [TC]² promised that, if an additional system were not yet available, they would loan theirs to CAR after the November 1992 ARC conference for a limited period of time to accomplish the requirements of Phase 2.

At the July 16 meeting, Jud Early gave Roy Pargas a diskette containing the data for the torso (neck to mid-thigh) of one male person (Figure 1). Dr. Pargas used the data to screen a number of potential computer graduate students who were interested in working on this project. Six students developed software to display the figure on a SUN workstation. The students with the best displays were selected in late August to work on the project. Dr. Pargas continued to survey the field for an alternative to the Dimensional Measurement Systems, Inc. 3D, non-contact body measuring equipment.

Size Prediction Expert System Development

Continued review of the state of the art revealed that the Remind® software tool by Cognitive Systems, Inc., Boston, MA, was still the most suitable for this project. One of its major features, induction, was missing in most other systems. Induction provides automatic processing of cases to classify them and construct a search tree. In the size prediction situation, the tool can process the hundreds of cases accumulated from activities at Fort Jackson's Clothing Initial Issue Point. Remind® can determine automatically which features are most relevant in discriminating among the clothing sizes. Thus induction expedites development of a prototype system.

Negotiations with Cognitive Systems resulted in a donation of the development system and C++ libraries. During July 1992 the project team completed the necessary paperwork, including a license agreement signed by a university representative and the forms necessary to register the donation officially and designate it as tax-deductible by the company.

The Cognitive Systems company later sent a newer version of Remind® (1.1) for use in this DLA project. One of the improvements was greater speed when running on IBM-compatible machines. Also they sent the C libraries, which allow developers to employ their own interface, as soon as they were completed. In the interim, the project proceeded using the basic Remind® tool, and was not delayed by waiting for the C libraries. All work with the basic tool was transferred to the C libraries later.

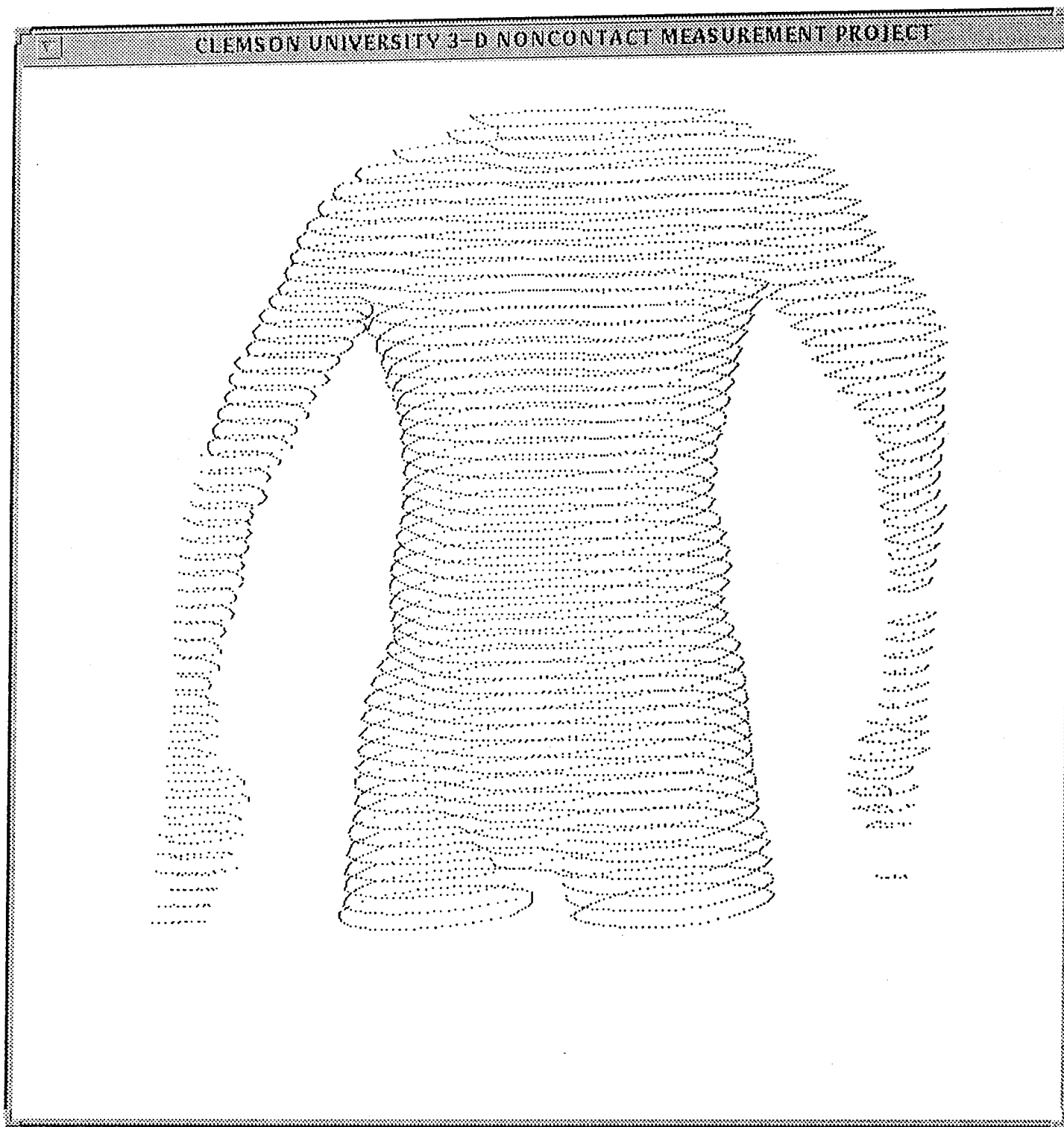


Figure 1. Example of body scan output.

On July 30-31, 1992 Dr. Steve Davis, Murali Earagolla, and Sarat Vemuri attended training sessions on the Remind® software tool at Cognitive Systems, Inc. in Boston. The training was certainly worthwhile and was directly relevant to important tasks in developing the system. It provided hands-on experience in using all major features of Remind®.

An explanation of case-based reasoning

As its name suggests, case-based reasoning simulates the human tendency in problem solving situations to compare and classify events according to previous experience. The level of human expertise in a given area is often related to the number of similar cases previously encountered. When seeking a "good" doctor or mechanic, one who has treated many cases which are similar to the given problem would generally be selected. In seeking an expert fitter of military clothing, the likely choice is a person at the CIIP who has fitted many soldiers.

Most current expert systems have been developed using "IF THEN" rules which are chained together to arrive at a conclusion (such as what garment size is appropriate). Although rule-based systems are helpful for solving some types of problems, this paradigm seems not as appropriate as CBR for the size prediction problem. Using CBR should reduce the "knowledge engineering" time which would otherwise be spent manually extracting a complete and consistent set of classification rules from an expert (in the size prediction environment, classification rules would include heuristics for assigning sizes to sets of measurements which are not covered by standard tables). Instead, using the CBR approach, representative cases are gathered and descriptive features determined. The software tools are then used to help determine the most important descriptive features for accomplishing the size prediction. For the size prediction situation, the Phase I descriptive features, the body measurements used in the current (manual) system, are known. With the incorporation of 3-D, non-contact measurement during Phase II, the possibility of using non-standard measurements as descriptive data will be explored.

To add knowledge to a rule-based system requires careful programming, but it is relatively easy to increase the knowledge of a CBR system by adding new cases. CBR systems have a better way to explain their reasoning than do rule-based systems. CBR systems can explain results both through describing the general rules involved in selecting a conclusion and also can show examples of previous real cases that are similar to the problem at hand. Rule-based systems can only report to the user the chain of rules that led to a conclusion. A rule-based system does not present examples or cases (even though their rules may be based on cases).

Preliminary System Plans

One of the most valuable lessons of the Remind® training was how to import cases (in this project each case consists of measurements and sizes of garments for a particular soldier). The CIIP at Fort Jackson provided several hundred cases a week, and the project team, assisted by volunteers from the Clemson Apparel Research staff, stored them in dBase IV files (1008 cases were stored during July 1992). This data is imported by converting the dBase files to standard character format. The Remind® tool can read files in that format if the various fields (such as neck size, sleeve length, etc.) are identified. The Remind® tool can automatically analyze the cases to determine which features seem most important in predicting sizes. Therefore, the plan was to employ that analysis initially to determine trends in the data. It is also possible to specify manually how the cases should be organized. Before the system can be used to predict sizes, the cases must be structured in a logical way so that the search for similar previous cases can proceed efficiently.

When cases have been entered and organized, the system can be tested by reserving a group of known cases as test input. For example, a set of measurement and size data for 100 soldiers could be used (where the sizes are known to be correct). Remind® predicts sizes based on the previous cases. If a high percentage of the tests produce correct predictions, the system can be considered ready to use. If too many predictions are incorrect, the stored cases are examined to determine if any are incorrect, whether additional cases should be stored, or if the stored cases should be reorganized so that the search proceeds in a different way. The testing can be conducted periodically as cases continue to be added to the system. Most likely the performance of the system (in terms of correctness of predictions) improve as cases are added, but at some point the performance increase begins to flatten. At that point the system has sufficient cases for further predictions.

On July 31, 1992 Dr. Steve Davis visited Dan deLuis at Natick R&D labs to coordinate progress of this project. Dan reaffirmed his interest in accompanying the project team when the first tests of the prototype system are conducted at Fort Jackson.

Coding Scheme Used for Recording Cases

In order to enter data from the soldier clothing forms into a database, a format was selected to be reasonably easy for those accomplishing the work. There are a total of 16 fields (data items) per record in the dBase IV file as follows.

LNAME—entered as it is on the clothing form.

FNAME—entered as it is on the clothing form.

HEIGHT—entered in inches, although it is expressed as feet and inches on the clothing forms (e.g. a height of 5'6" is entered as 66).

HEAD—entered as it is on the clothing form, except when there is a fraction it is entered as a decimal value (e.g. if head=23 1/4 it is entered as 23.25).

NECK, CHEST, WAIST, HIPS & SLEEVE LENGTH—the same convention as for HEAD is followed.

WEIGHT—entered as it is on the clothing form.

SHIRT_SHOR—this is taken from the "short sleeve" item on the clothing form. Some interpretation of this item is necessary. On the clothing form it is represented by either a 2 digit number or a 3 digit number. If it is a 2 digit number it is to be entered as it is into the dBase file. If it is a 3 digit number on the clothing form, the 3rd digit will always be a '2' and should be entered as "h" (e.g. 152 is entered as 15h; 162 is entered as 16h and so on).

SHIRT_LONG—this is taken from the "long sleeve" item on the clothing form. It is represented as shirt-size by sleeve-length. Thus, for example, 15X30 is to be entered as 1530. The 'X' mark is never entered. If there are three digits before the X, the last of them will be a 2 and it is to be entered as "h". For example, 152X30 is to be entered as 15h30. Additional examples are given below.

Clothing Form Entry	dBase File Entry
16X32	1632
172X34	17h34
14X28	1428
162X30	16h30

CAP—entered as it is on the clothing form. (it could be a single digit number or a 3 digit number in the case of integers plus fractions). Care is taken not to enter the next field on the clothing form, "gloves," by mistake.

COAT_BLACK—this is taken from the "coat all weather" item on the clothing form. The first 2 numbers are entered as they are. The alphabetic suffix is entered as it is, except "xs" is to be entered as "y", and "xl" is to be entered as "x". Examples are given below.

Clothing Form Entry	dBase File Entry
36s	36s
34r	34r
32l	32l
30xs	30y
37xl	37x

COAT_GREEN—this is taken from the "coat AS AG 344" item on the clothing form. The same convention as for the COAT_BLACK is followed.

TROUSERS—the same convention as for the COAT_BLACK is followed.

If any values are missing or unreadable or obviously in error (unreasonable), data from the entire clothing form is be discarded.

Progress of Data Entry

Data entry continued during August, 1992 and the database grew to over 2500 records. Each record represents data from the clothing record of a soldier. The format for initial data entry was selected earlier in the project to facilitate speed of entry. Later, the project team determined that a different format would be more appropriate for use in the size prediction system. The team developed a program (TRANSFER) which takes a source file in the original format and produces a file in the new format. It converts certain fields to consistent formats. For example, it converts all alphabetic codes to uppercase, and converts shorthand notation to numeric form. For example, a "15H33" entry (which was entered on the clothing form for the long sleeve shirt) is converted to two fields. The first field has 15.5, (the neck size) and the second has 33 (the sleeve size). Another example is that the cap size entered as 718 will be converted to 7.125. The structure of the new database is as follows:

<u>Field</u>	<u>Field Name</u>	<u>Type</u>	<u>Width</u>	<u>Dec</u>
1	LNAME	Character	1	15
2	FNAME	Character	1	15
3	HEIGHT	Numeric	6	2
4	HEAD	Numeric	6	2
5	NECK	Numeric	6	2
6	CHEST	Numeric	6	2
7	WAIST	Numeric	6	2
8	HIPS	Numeric	6	2
9	SLEEVE	Numeric	6	2
10	WEIGHT	Numeric	6	2
11	SHIRT_SHOR	Numeric	4	2
12	SHIRTLONG1	Numeric	4	(neck size)
13	SHIRTLONG2	Numeric	3	(sleeve size)
14	CAP	Numeric	6	3
15	COATBLACK1	Numeric	4	(size)
16	COATBLACK2	Character	1	(length)
17	COATGREEN1	Numeric	4	(size)
18	COATGREEN2	Character	1	(length)
19	TROUSERS1	Numeric	4	(size)
20	TROUSERS2	Character	1	(length)
21	DATE_ENTER	Date	8	(date of issue)

The codes for lengths are as follows:

Code	Size
Y	XS
S	S
R	R
L	L
X	XL

Checking for Validity of Cases

The team developed a set of programs which help identify invalid cases. One checks measurements within reasonable ranges (MEASCHK). Another checks that sizes are within reasonable ranges (SIZCHK). SIZMEAS checks correspondence between measurements and sizes. For example, a case is flagged for further examination if the sleeve length differs from the arm length by more than 2 inches. SSNECK and LSNECK check reasonable correspondence between neck measurement and neck sizes of short sleeve and long sleeve shirts. COAT checks the correspondence between sizes of the black coat and green coat to see if they are within a reasonable range. After correcting all mistakes in data entry, the next step is to examine all "strange" cases. If it is determined that the data on the clothing form was probably erroneous, the case is deleted from the database, because the success in predicting sizes depends on realistic data in the database.

Determining How to Handle Cases in Remind®

Coat, trouser, and sleeve lengths are stored in separate fields in the database. The sleeve length is numeric, and the others are symbolic (letter suffixes such as the R in 36R). Since the letter suffixes are in a separate field, an ordering can be defined using the symbol editor. The initial plan was to predict length separately from size. For example, there would be a computer run to determine the size of the green coat, and another run to determine the coat length (regular, long, extra long, etc.). The rationale for this approach was that the lengths are probably predicted in a different way than the basic size. For example, intuitively height is one of the main predictors of length, but other measurements may be more important in predicting size. In Remind® terminology, there are two "outcome" fields for certain garment sizes. For each outcome field, there must be established a separate "cluster" (search tree) which will be searched at run time. The first approach used was to predict length separately from basic size. Apparently it does not make sense in a case-based reasoning tool such as Remind® to designate more than one "outcome" field (e.g. long-sleeve-shirt neck size and sleeve length.) To predict size and length at the same time, they must be coded in a single field. One possible single-field coding option is to convert

coat/trouser suffixes to decimal values. e.g. 36S could be 36.0, 36R could be 36.2, etc. But this approach might cause a problem in that 35.9 would be interpreted as closer to 36.0 than would 36.2, for example.

The next plan was to predict size based on a retrieval of a number of similar cases, and then select length from those retrieved cases. The rationale for this idea is that the correct length should be reflected in the retrieved cases even though it was not the basis for the search, and this approach saves the time of an additional search.

Run Time for Predictions

It became apparent that the prototype system might not run fast enough for operational use. A few sample runs were made to predict the size of the short sleeve shirt. Even with a relatively small database of cases, the prediction took about 30 seconds on an IBM PS/2 Model 70 with 8 megabytes of memory. Run time can be improved with a faster computer, for example a PC with 486 processor. Also, run time was expected to improve when C libraries were used rather than the interactive version of Remind®.

Progress of Expert System Data Entry

Data entry continued during September 1992 and the dBase IV database grew to over 3500 records. Each record represents data from the clothing record of one soldier. Many of the clothing forms received in August reflected issue of new models of the long sleeve shirt. Sleeve lengths were combined in many cases (e.g., one of the new lengths is 32/33). The project team decided to enter the larger of the two sizes in the database.

The First Prototype Prediction System

To predict the size of a given type of garment, such as the short sleeve shirt, requires establishing a "cluster" (a search tree) for that particular garment. Each cluster takes considerable computer time, about an hour, but this process only has to be done once for each garment type. During September 1992, clusters were established for the short sleeve shirt, long sleeve shirt, black coat, green coat, and trousers. These clusters were based on 2500 cases.

Testing of the system was in the early stages during late September. It took some work to develop tests that provided meaningful results. The Remind® software tool has a built-in testing mechanism, but because it normalizes all data (including garment sizes) the results require careful interpretation. For example, if the testing results for 100 predictions indicate that 75 of them were rated at "90 to 100 percent similar" to the correct size, it is not obvious what size corresponds to "90 percent similar".

Preliminary testing revealed instances of several soldiers who have identical body measurements but were assigned different sized garments during clothing issue. These instances could occur for various reasons, including garment size variances or differences in body shapes which are not reflected by the standard measurements. A strategy is required to address this situation when predicting garment sizes for a new soldier whose measurements match a set like the aforementioned. Initially the plan for such instances was to have the computer system provide multiple ranked recommendations, (e.g., "The most likely size is 39R; the next most likely size is 38L," etc.). The ranking could be based on frequency of occurrence of size assignments in the previous cases.

Raktim Sen, a graduate student in Dr. Davis' fall class, in order to satisfy the requirements of a course project, volunteered to help develop a testing strategy. He attended weekly meetings of the team and studied the software.

Demonstration at Bobbin Show

A prototype size prediction system was demonstrated at the Clemson Apparel Research booth at the 1992 Bobbin Show. All visitors who viewed it seemed favorably impressed with the project. Selected volunteers who knew their proper shirt size were asked to participate in a test of the accuracy with which the system could predict their short sleeve shirt size. They were measured for neck and chest size, and were asked to tell their height and weight. Those values were input to the computer system which then predicted the shirt size. For all of the volunteers, the prediction was correct.

During the month of October 1992, data entry continued. The team employed programs to ensure accuracy of the clothing records which were entered into the dBase IV database, now containing over 4000 records.

Improved Prototype Prediction System

The system was fully functional (in terms of selecting sizes) by late October. The two top-priority tasks were to evaluate the size prediction accuracy and to enhance the prototype such that it could be effectively used at a Clothing Initial Issue Facility. Speed was to be improved if possible. The first system was rather slow, requiring about 30 seconds to predict the size of one clothing item. It was anticipated that the speed would improve if C libraries were used instead of the interactive version of Remind®. Cognitive Systems indicated that those libraries would be available in the near future. Clemson University needed to sign a separate license agreement for the new libraries. Another approach to speed-up was suggested by Sarat Vemuri. He pointed out the possibility of converting the case-based reasoning system to a rule-based system. It appeared that such a system could be efficiently coded to speed up the decision process. He developed software tools to help accomplish such a

conversion of the system. Unlike a rule-based system that would have been created without Remind®, this system employs the rules determined by the case-based-reasoning. The resulting rules reflect actual case size assignment rather than intended size assignment by body dimensions.

Sarat Vemuri employed Unix system tools (Lex and YACC) to develop a program which converts a description of case indexing to C language IF statements which implement equivalent rules. Unfortunately, the description of the case indexing may have to be entered in the computer manually. Even though the index structure is produced by Remind® in machine-readable form, Remind® only provides access to it through the computer display. Although the aforementioned approach proved feasible, it was not pursued further. It seemed more practical to achieve satisfactory speed by employing a faster computer with the C libraries.

The project team completed license agreements for an advance copy of the C libraries for the Remind® case-based reasoning tool. These libraries are necessary to enhance the prototype such that it can be effectively used at a Clothing Initial Issue Facility because, although the interactive version of Remind® is sufficient to test the size prediction capability, it does not allow for developing an interface which would be suitable for users at a CIIP. Cognitive Systems, Inc. sent copies of the libraries, but after many hours of work the team was unable to install them on a computer. There were some basic problems with the version of C that had been used to produce the libraries. After hearing about our problems, the company acknowledged that changes were necessary and they sent a revised version.

System Predictions

There are several ways of designing the ReMind® system to make a prediction. The project team had to postpone making a decision until after a satisfactory testing procedure was developed. Given a test case, the system can automatically retrieve, according to the index hierarchy, a group containing a minimum of "n" most similar cases from the database. If the outcomes of all the cases are the same, the prediction is obvious. But there must be a strategy to handle situations in which the outcomes are mixed. An obvious scheme is to take a majority vote (or make an arbitrary choice in case of a tie). Another method is to perform a "nearest neighbor" search of the n cases. This approach involves deciding on the weighting of the match fields (the body measurements). Alternatives include assigning equal weight to all match fields and selecting weights according to priorities established in the index hierarchy.

In the techniques described above, the built-in procedure to retrieve a minimum of "n" similar cases might not be the best way to obtain a group of cases if the prediction will be determined by voting. In some situations the similarity of the cases of the group could vary significantly, and the prediction might be determined by a majority of cases which are much less

similar to the new case than are another (smaller) subset of retrieved cases whose outcomes agree. Therefore it may be desirable, prior to the vote, to eliminate from the group those cases whose similarity falls below a specified threshold.

The aforementioned methods may work well for instances when there are a sufficient number of similar cases. But there are relatively few cases for individuals who are unusually large or small or who have unusual proportions, so the size predictions for such individuals tend to lack a sound foundation and may very well be wrong. The system should at least provide a warning for such situations. The warning could be triggered when the similarity scores of the retrieved cases fall below a certain threshold. Unusual individuals could then be fitted manually. Better yet, the system could be equipped with rules which adapt the outcome of the most similar retrieved case. An example of such a rule is:

```
"IF retrieved_arm_length > problem_arm_length + 2  
THEN predicted_sleeve_length = predicted_sleeve_length +  
    (retrieved_arm_length -  
    problem_arm_length)."
```

Such a system would be a hybrid, part CBR and part rule-based. Rules could also be used to help identify people who cannot be fitted with a standard size, which would avoid needlessly trying on garments.

There are no well-accepted theories or guidelines to determine which of the alternative strategies would lead to the best performance. Therefore, empirical investigation was the basis for designing the prediction strategy.

Given a set of body measurements, the case-based reasoning tool retrieves the most similar cases from the database. Since the clothing sizes of the retrieved cases may not all be the same, a scheme is necessary for the system to recommend a particular size. The first step for the prediction alternatives was to retrieve inductively a minimum of 20 cases. The project team explored several possibilities for selecting a specific recommended size:

- a. Take a vote among the group of retrieved cases.
- b. Determine the similarity scores of the retrieved cases, according to a "match field weights," then choose the size associated with the case having the highest similarity score.
- c. Select the n most similar cases from those described in b above and take a vote among them (currently n = 20).

The weight vector mentioned in b involves assignment of weights to the match fields involved in computing the similarity score. The match fields are those deemed to be important in determining size. There is no prescribed way of selecting the fields or their weights. For the long sleeve shirt, the

match fields were selected based on the project team's knowledge and intuition. These were chest, height, neck, sleeve, and weight. Selection of field weights was based on an examination of the inductive search tree. Fields associated with branches near the root of the tree were considered to be more important, and thus they were assigned higher weights. For example, for the long sleeve shirt the weights were: weight 16 (36%); chest, height and neck 8 (18%), and sleeve 4 (9%). The project team experimented with an algorithm for selecting the field weights, but decided the determination should be subjective.

The scheme mentioned in c above seemed to be the best, at least for determining long sleeve shirt size.

Improved Prediction Scheme

Work continued toward finding a good prediction scheme and on determining a reasonable way to evaluate system performance. Attention was focused on the long sleeve shirt. The team initially used as test cases the records of 51 soldiers who had been randomly selected during a visit to Fort Jackson, SC. Measurements and garment sizes had been checked by Dr. Staples; she took precise measurements, whereas the fitters normally rounded up to the nearest inch. Initially the tests employed her precise measurements, but then the decision was to employ the fitters' measurements instead, because results would be more realistic.

The first prediction scheme tested involved inductive retrieval of a minimum of 20 cases. Then a vote was taken among the 10 of those cases which had the best similarity scores, according to "match field weights." The size receiving the most votes was designated as "first choice," the one receiving the next highest number of votes was called "second choice," and so on. Ties were resolved by computing the average similarity score. Match field weights were selected by examining the inductive search tree. Fields associated with branches near the root of the tree were considered to be more important, and thus they were assigned higher weights.

The inductive retrieval, the first step in the prediction process, involved an index (search tree) based only on the garment neck size; the sleeve size was ignored. Likewise, the computation of similarity scores was based on criteria thought to be relevant to the prediction of garment neck size alone. Thus, the sleeve size was a kind of "tag-along."

Using the aforementioned scheme, the prediction rate was not very good. The first choice of the system was correct, in terms of garment neck and sleeve size, for only 13 of the 51 cases. Therefore the project team decided to try a different prediction scheme. The new approach involved separately predicting garment neck size and garment sleeve size, each using a different index (search tree) and a different assignment of weights for the computation of similarity score.

The revised versions of the C-libraries for the Remind® case-based reasoning tool arrived during February 1993 and they began to work properly. The first item on the agenda was to find a way to automate the testing process, which formerly was very labor intensive. Successful automated tests were performed for two garments, short sleeve shirt and long sleeve shirt. For 120 randomly selected test cases, the first choice of the system for short sleeve shirt size agreed with the issued size for 67% of the cases, and the second choice agreed in 31% of the cases. For the long sleeve shirt the first choice of the system agreed with the issued size for 56% of the cases, and the second choice agreed in 25% of the cases. Thus for these tests the system was not as accurate as the fitters at Fort Jackson. The team was not satisfied with the performance of those tests, in terms of accuracy of predictions, but they provide a basis for experimentation to calibrate the prediction process.

Work continued toward finding a good prediction scheme. Following successful use of the revised versions of the C-libraries for the Remind® case-based reasoning tool to conduct automated tests, the team was still not satisfied with the performance of those tests, in terms of accuracy of predictions. The automated tests, however, do provide a basis for experimentation to calibrate the prediction process. A number of weight-assignment schemes were tried for conducting nearest-neighbor retrievals (following inductive retrieval), but for the long sleeve shirt none was able to achieve "correct" predictions above 60% ("correct" is defined as predicting the same size that was actually issued).

There are two situations involving incorrect predictions.

- 1) The system strongly votes for a particular size, but it is "wrong" (e.g. 8 of 10 soldiers with very similar measurements were issued size 15, but the test case has size 15.5).
- 2) The system is "not sure" about the size (e.g. of the 10 soldiers with the most similar measurements in the database, 3 were issued size 15, 3 were issued size 15.5, 3 were issued size 16, and 1 was issued size 16.5).

Without being able to examine the actual garments and soldiers involved, it is impossible to be certain of an explanation for the wrong predictions. Possibilities include:

- 1) For situation 1 above, the garment size selected by the system could actually be as good or better than the size issued.
- 2) For situation 1 above, the test case could involve an individual with an unusual shape, but the unusual characteristics are not reflected by the body measurements.
- 3) For situation 2 above, the body measurements could represent a soldier who is not well fitted by any of the standard sizes (is "in between" the standard

garment sizes), and the issued size will depend significantly on the judgment of the fitter.

Of course, it is also possible that the test case is erroneous in some way, such as: incorrect information entered on the clothing form, or the garment could have been mislabelled. The only way to be sure about the data would be to conduct a field test at Fort Jackson to help determine the cause of "wrong" system predictions. For every "wrong" prediction the team could examine the actual garment and soldier involved and eventually clear up the mystery.

Improved Data Import

Efforts next focused on improvement of the data import facilities and development of an operational system. Data import is involved in transferring data from the dBase IV database of soldier clothing forms to the case-based-reasoning tool. It must be performed often during system development activities. Data import is problematic because the data must be in precisely the right format. The dBase files have to be converted to ASCII (DOS text) format with some special character, such as the comma, separating each field. The dBase file was formerly converted to a text file and then a word processor was used to add the commas between fields. Previously data import had failed for reasons such as a line-feed character appearing in records (having been inserted automatically by the word processor). Therefore, the project team decided to undertake an extensive study to streamline and improve the process. It turned out that most problems were caused by the use of a word processor. Therefore, the team developed a method which produces the necessary text file directly from dBase IV. This was accomplished by adding extra fields to the dBase file to contain the separator character (the comma). Then the text file could be produced by a single command in dBase IV: COPY TO filename TYPE SDF.

During the work on data import, the file layout was also made more concise to eliminate storage of unnecessary data. The new format requires only 69 characters (not counting the separator fields which are not imported into the case-based reasoning tool); the old one required 120. This not only saves storage space but helps improve system performance as well. The new layout also facilitates easier reading by a human, because when it is converted to text format all the fields are a constant width. Columns of data are neatly lined up when printed or viewed on a computer screen. The new structure is shown in Table 1. In the future the format could be made even shorter. For example, the fields for soldier name could be eliminated.

Progress continued toward an operational system with a convenient user interface that would take soldier measurements as input and print a list of recommended garment sizes. Sarat Vemuri, the previously-employed student now working in Atlanta, contributed as a volunteer to keep this part of the project moving forward while graduate students Prahlad Yerra and Murali Earagolla were on vacation.

Table 1. Structure for database: A:\IMPORT.DBF

Field	Field Name	Type	Width	Dec	Index
1	LNAME	Character	3		N
2	B1	Character	1		N
3	FNAME	Character	3		N
4	B2	Character	1		N
5	HEIGHT	Numeric	5	2	N
6	B3	Character	1		N
7	HEAD	Numeric	5	2	N
8	B4	Character	1		N
9	NECK	Numeric	5	2	N
10	B5	Character	1		N
11	CHEST	Numeric	5	2	N
12	B6	Character	1		N
13	WAIST	Numeric	5	2	N
14	B7	Character	1		N
15	HIPS	Numeric	5	2	N
16	B8	Character	1		N
17	SLEEVE	Numeric	5	2	N
18	B9	Character	1		N
19	WEIGHT	Numeric	3		N
20	B10	Character	1		N
21	SHIRT_SHOR	Numeric	4	1	N
22	B11	Character	1		N
23	SHIRTLONG1	Numeric	4	1	N
24	B12	Character	1		N
25	SHIRTLONG2	Numeric	2		N
26	B13	Character	1		N
27	CAP	Numeric	5	3	N
28	B14	Character	1		N
29	COATBLACK1	Numeric	2		N
30	B15	Character	1		N
31	COATBLACK2	Character	1		N
32	B16	Character	1		N
33	COATGREEN1	Numeric	2		N
34	B17	Character	1		N
35	COATGREEN2	Character	1		N
36	B18	Character	1		N
37	TROUSERS1	Numeric	2		N
38	B19	Character	1		N
39	TROUSERS2	Character	1		N
40	B20	Character	1		N
Total			89		

Arrangements for Continued Use of the Case-Based Reasoning Tool

The project team negotiated a donation of Remind® from Cognitive Systems, Inc. at the beginning of the project. Continued use of Remind® required payment of maintenance and support fees. Fortunately, Clemson Apparel Research (CAR) agreed to support that cost with Clemson University funds. The project needed a second copy of the software to facilitate use by the graduate assistants and Dr. Davis. One copy was installed at CAR and another on campus. The second copy was also useful when field tests were conducted at Fort Jackson, so that the software development could continue during the tests.

To purchase the second copy, Dr. Davis devoted \$1800 from the Management Department fund intended to support in-house teaching and research expenses. He also negotiated with Cognitive Systems to obtain additional software licenses to allow teaching Remind® to students in his graduate class on Management Support Systems. Involving Remind® in classes helped the project in two ways: it helped deepen Davis' understanding of the tool and allowed students in the class to contribute ideas on the project. The company agreed to provide 23 additional software licenses to allow teaching Remind® to students at Clemson University.

Review and Project Planning

Several meetings of the project team were held during July 1993 to review the status of the project and plans for future efforts. Two important issues were how to pursue obtaining 3D measuring equipment and how to get the most value from this project if the 3D equipment did not materialize.

The expert system component can function without 3D equipment, but its performance is limited by the relatively few body measurements available, and by inaccuracies in the data previously obtained from clothing records. One of the alternatives discussed was to visit Fort Jackson to obtain more accurate measurements (since the fitters round up all circumference amounts to the next whole number, thus the measurements recorded on the completed forms provided by Fort Jackson are not precise). Such an effort would be very labor-intensive, and still would not guarantee significant improvement in system performance. Therefore the decision was not to pursue that course of action. Instead, work continued on developing a prototype system which could be tested at Fort Jackson. Even though the performance would probably not be better than human fitters, such a test could help determine the feasibility of employing this kind of system in an operational environment. Most of the work to date had focused on experimenting with predictions of short sleeve shirt size. Now the system was calibrated for predicting sizes of the other garments.

A Field-Test-Worthy Prototype Expert System

Graduate students Prahlad Yerra and Sarat Vemuri worked independently on developing a prototype system which could be tested at Fort Jackson. After taking a full-time job in Atlanta, Sarat Vemuri continued to work on this project as part of his M.S. thesis. He visited Clemson several times to demonstrate his system and get guidance on his work.

Prahlad's system works as follows. Take input constituting one soldier's measurements from a file. Define a new case, based on those measurements. Repeat the following for every view (there is a view for each garment size which is to be predicted).

- (1) Open the view with a selected weight vector.
- (2) Inductively retrieve a minimum of 20 cases.
- (3) Compute the nearest neighbor "similarity" score for the retrieved cases.
- (4) Take a vote among the retrieved cases (previously the system just selected the case with the highest similarity score).

The size which wins the vote is the size predicted by the system.

One of the tedious aspects of writing the programs was the need to convert data to and from the internal format used by the C libraries of the Remind® case-based reasoning tool. Prahlad tested his system with the short sleeve shirt and did not set up views for the other garments. Although a view can be constructed automatically by the Remind® case-based tool, human judgment is necessary in setting certain parameters and is involved in determining which variation of a view is best.

The project team tentatively decided to employ a separate search for the size/length for those garments which have both. Therefore the system would require separate views for green coat, green coat length, black coat, black coat length, short sleeve shirt, long sleeve shirt size, long sleeve shirt sleeve length, trouser size, trouser length, and cap size.

Still to be determined was whether the system required a way to browse stored cases interactively, and whether it needed a built-in way to store new cases as they were input. Once actual sizes are assigned to a soldier, a new case is available.

Setting up a library for each garment was tedious. The Remind® software tool builds a single file containing all clusters. Perhaps because of the size and complexity of the file, a number of unresolvable errors occurred during cluster building. After an error occurrence during cluster building, the only option was to re-start another 2 1/2 hour process. To reduce the amount of storage required for the cases, the structure of the database was made more concise; only essential information was retained, and the size of the fields was reduced to the minimum necessary. For example, soldier names were

eliminated, because they are not involved in size prediction. The resulting file is still quite large, over 4 MB, but is smaller than it would otherwise be. This reduction in file size seemed to reduce error occurrences. Only one clustering produced an error.

Afterward, the team successfully developed clusters (indexes) for the case-based expert system for a 4000 record database to allow prediction of the following: short sleeve shirt size, long sleeve shirt size, long sleeve shirt sleeve length, green coat size, green coat length, black coat size, black coat length, trouser size, trouser length. Not counting set-up time, each of the aforementioned clusters took about 2 1/2 hours processing on an IBM-compatible 386 PC.

After the first rough prototype system was developed, programmers Murali Earagolla and Prahlad Yerra left the project to take jobs in Florida. Sarat Vemuri continued to work part time on the project. He developed prototype software written in Borland C++ for Windows which provided input/output to the expert system. It took a set of body measurements as input, consulted the expert system to obtain predictions, and displayed predicted sizes. Using the 4000 record database on an IBM-compatible 386 PC it took about 20 seconds per prediction, which amounts to between 1 and 2 minutes for all predictions for one individual. This was satisfactory for an operational test. A higher speed could easily be achieved by using a 486 PC.

The next step was to improve the user-friendliness of the system and to add a capability to print results. The team also experimented with various weightings of the importance of the body measurements to determine what fosters the most accurate size prediction. An operational test at Fort Jackson was planned for November 1993.

An outline of the plans were sent for review to Lonnie Turner, manager of the Clothing Initial Issue Point at Fort Jackson, SC. The main objective was to test the concept of automated size prediction while avoiding any significant disruption of operations at the CIIP. The computer system and laser printer were to be stationed in the measurement-taking area of the clothing issue facility. The team was to take an extra computer for backup. One of the team was to operate the computer to input soldier measurements into the computer as they were called out by the measurer. Predictions were to be printed and given to each soldier by stapling a paper on top of the clothing form. The paper would indicate 1st, 2nd and 3rd choice for each garment. Fitters were to be asked to try garments in accordance with the computer predictions. At the end of the line, one of the CAR team was to record the size actually assigned on the CAR form and retain the CAR form.

For the expert system, the team considered various weightings of body measurements to help improve accuracy of predictions. Of course, the possible accuracy of the prediction system was limited by the accuracy of the data upon which it is based. Initial weightings are shown in Table 2.

First Operational Test

On November 10, 1993, Dr. Davis performed the first operational test of the size prediction system at Fort Jackson, S.C. The purposes of the test were to verify the feasibility of real time computer-based size prediction at a military clothing issue facility and to determine how the computer system could be smoothly integrated into operations of the facility. Mr. Lonnie Turner of the Clothing Initial Issue Point was extremely supportive and provided everything necessary to conduct as thorough a test as possible at this stage of the project.

Two computer systems were stationed on a table at the back of the fitting room during the evening before the test. One of the systems served as a back-up. On the day of the test, recruits initially sat on benches in the fitting room. The fitter measured each of them at the front of the room. The fitter called out the measurements to a soldier on detail who copied the data onto a clothing form and then put it on a clipboard carried by the soldier. After the measuring, soldiers walked to the computer table. They told Dr. Davis their name and measurements and he entered them in the computer. In order to avoid slowing down the operation, he asked soldiers to bypass the computer station if he was not ready for them (a total of 71 soldiers were processed). Soldiers put the size prediction page on top of the clipboard. Fitters followed the recommended sizes for try on, and listed on the size prediction sheet any other sizes tried or issued. Soldiers returned to the fitting room after getting their garments, where a detail soldier collected the size prediction forms.

Overall, the operational test was quite successful. This experience showed that automated size prediction can be easily integrated into current operations. No significant problems occurred.

The size prediction procedure would be improved by entering the measurements into the computer as they are called out by the fitter. This would reduce the chance of error in transcribing, would not add any significant time to the existing process, would provide more legible data on the clothing form, and would avoid an extra sheet with size predictions. This could be accomplished if the computer were stationed beside the fitter. The computer could print the predicted sizes, together with the customary data, directly on the clothing form. The computer could be operated by the detail soldier (the one who currently records information manually on the clothing form). Consideration was given to revising the software to allow testing the system in the aforementioned way. Options for printing predicted sizes directly on the clothing form were discussed with Mr. Turner.

Table 2. Initial Weightings

Short or Long Sleeve Shirt Size			
Chest	8	Sleeve	0
Height	1	Waist	1
Hips	0	Weight	16
Neck	16	Head	0
Long Sleeve Shirt Sleeve Length			
Chest	1	Sleeve	16
Height	8	Waist	1
Hips	0	Weight	16
Neck	4	Head	0
Black or Green Coat Size			
Chest	8	Sleeve	0
Height	1	Waist	1
Hips	0	Weight	16
Neck	16	Head	0
Black Coat Length			
Chest	4	Sleeve	8
Height	16	Waist	1
Hips	8	Weight	16
Neck	0	Head	0
Green Coat Length			
Chest	2	Sleeve	16
Height	16	Waist	2
Hips	2	Weight	8
Neck	2	Head	0
Trousers			
Chest	0	Sleeve	0
Height	0	Waist	16
Hips	16	Weight	8
Neck	16	Head	0
Trousers Length			
Chest	0	Sleeve	0
Height	16	Waist	8
Hips	8	Weight	8
Neck	0	Head	0

It took some time to analyze the results and determine how accurate the predictions were. Checking system accuracy was not the main purpose of the operational test, because the system had not yet been calibrated for maximum accuracy. Calibration involves selecting a test sample and empirically determining which weightings of measurements work best for determining similarity of bodies. This is a tedious, time-consuming process requiring multiple computer runs. The soldiers processed on the day of the operational test served as a suitable test sample.

The test revealed several opportunities to improve the size prediction software.

1. The measurement fields should be in the same order that they appear on the clothing form, if the data is to be transcribed from the form as was done during this test (it was awkward to enter data in a different order). If measurements are going to be recorded as they are called out by the fitter, then the fields should be in the order they are taken by the fitter: height, head, neck, chest, waist, hips, sleeve, weight.
2. A laser printer might work better than the dot matrix printer used in this test. With the dot matrix, the computer operator has to wait until a page is printed to tear it off and give it to the recruit (it's a bit tricky to tear off the page without messing up the alignment of the paper feed). With a laser printer, each recruit could pick up his own form. The computer operator could go on to process another recruit after giving the print command.
3. A "predict and print and clear" button would save time. At the time of this test the computer operator had to select the "print" button after each prediction, then "OK" and then "clear."
4. The printed form could be streamlined. The Clemson University logo should be eliminated. The measurements should be listed in the same arrangement as on the clothing form (sleeve and waist in right column). The data could be put in a smaller space so that it could be cut and pasted in the unused, lower right part of the clothing form.
5. After OK and Clear, the cursor seems to vanish. Apparently the user has to press the TAB key to get the cursor at "first name." This is not a big thing and should stay the way it is if it would complicate the program.
6. On the 486 machine, after four predictions the data input window began to display distorted labels and boxes; the distortion got progressively worse with continued use. The problem could be remedied by closing and restarting the program. Other windows (for example the results window) were not distorted. This was apparently a hardware problem of the 486 machine.

7. On the 486 machine, on three occasions the software crashed with the message "PROGMAN caused a general protection fault in module USER.EXE at 000F:1AB1." This was apparently a problem of this particular computer.
8. The printed output seemed to "creep" toward the right side of the page on the dot matrix printer. i.e. the left margin seemed to shift toward the right on successive printed outputs (this was not caused by lateral shifting of the paper.) This creep caused no problem during the test, but could be a problem if an attempt were made to print on a special narrow paper.

Prediction accuracy for the operational test at Fort Jackson on November 10, 1993 is summarized in Table 3. Seventy soldiers were processed by the expert system. Those soldiers were given computer-printed output which listed predicted sizes for garments. Clothing issuers could consider system recommendations when selecting garments for try-on. At the end of the test, the expert system accuracy was computed by considering the issued size to be correct. For a given garment, the percentages for first, second and third choice may sum to more than 100% because in some cases the recommendations for 1st, 2d, and 3d choice are not distinct.

Table 3. Accuracy percentages from test at Fort Jackson, November 10, 1993, for 70 soldiers (overall garment).

Code	Garment	Choice number	Percent accurate
SSS1	Short Sleeved Shirt, size	1	47.1
SSS2	Short Sleeved Shirt, size	2	30.0
SSS3	Short Sleeved Shirt, size	3	27.1
LS_1	Long Sleeved Shirt, neck and sleeve	1	27.1
LS_2	Long Sleeved Shirt, neck and sleeve	2	15.7
LS_3	Long Sleeved Shirt, neck and sleeve	3	18.6
BC_1	Black coat, size and length	1	47.1
BC_2	Black coat, size and length	2	12.9
BC_3	Black coat, size and length	3	20.0
GC_1	Green coat, size and length	1	38.6
GC_2	Green coat, size and length	2	12.9
GC_3	Green coat, size and length	3	15.7
TR_1	Trousers, size and length	1	31.4
TR_2	Trousers, size and length	2	11.4
TR_3	Trousers, size and length	3	7.1

To analyze the accuracy of size predictions which took place during the operational test at Fort Jackson, the test data was manually entered into the computer. This included entering the data from soldiers' clothing forms. Since manually analyzing the data would be tedious and error prone, the work was done by computer. However, the programs which run the tests could be modified in the future to facilitate automated testing.

Related Issues

Dr. Davis accompanied Dr. Chris Jarvis and Dr. Jack Peck on a trip to Fort Jackson during December 1993 to coordinate several research projects. A meeting was held with Doug Burchette and several others associated with clothing issue. He was very interested in the progress of the 3D project, both in the potential benefits of size prediction and later of automating made-to-measure garment development.

The meeting included a visit to the clothing issue facility and a tour of the warehouse. Mr. Burchette was interested in automating the accounting of clothing issue and integrating it with an inventory system. Currently those issuing clothing record information manually on a clothing form. Later, data on the clothing form is manually entered into a computer and then reprinted. The computer-printed form is signed by the soldier and entered in the military record. The computer summarizes issues for each day. That summary is manually entered in a different computer which transmits the data to the Defense Personnel Supply Center.

Mr. Burchette envisioned an improved system which would record clothing issues directly into a computer, perhaps using barcode technology. This could improve accuracy and reduce labor requirements.

This visit was helpful in visualizing how an automated measurement system would fit into the clothing issue operation. Initially the system could improve operations by printing body measurements and predicted sizes directly on the clothing form. Later, when the clothing issue operations are automated, the measurement data could go directly into the computer.

Continued Expert System Performance Evaluation

Work continued toward evaluating performance of the expert system. The best way to check performance was to select test cases (soldier clothing records which include the sizes of clothing selected by human fitters) and then have the computer automatically compare the predictions of the expert system to the sizes actually issued. The assumption was that the sizes issued are correct. However, since some improvements had been made to the expert system, the programs which ran the tests needed to be modified.

Sarat Vemuri continued to contribute to the project by improving upon the software for the expert system. He successfully revised a batch testing program which allows running a large number of tests without human intervention.

The project team obtained from Cognitive Systems, Inc., the latest version of the Remind® application program interface software. The team provided it to Sarat and asked him to upgrade the expert system to work with the new version.

The project team found a problem in the testing program, which seemed to work well in testing performance in predicting short sleeve shirt size, but did not handle the other garments. The testing program should identify "hypothetical" cases (the test cases) in the database, perform a prediction for each such case, and compare the results with the size given in the "hypothetical" cases. For other than the short sleeve shirt, the program was not identifying the proper cases as hypothetical. The team worked on resolving this problem.

Second Operational Test

Plans were made for Dr. Davis to travel again to Fort Jackson to conduct an operational test of the revised expert system for size prediction. The objective was primarily to streamline the logistics of placing the expert system and its output in the current flow of uniform issue for soldiers.

Steve Davis traveled to Fort Jackson to conduct the operational test of the expert system for size prediction on June 8, 1994. The initial plan was to set up a computer table close to the place where soldiers are measured, just downstream of the flow. At that location the computer operator could listen to the fitter calling out measurements and enter them directly into the computer. This plan was revised because there was an especially large group of soldiers to process, and the measurements were taken by three detail soldiers working simultaneously (note the potential error of using detail soldiers for measurement taking, not preferred, but too often the reality). It was not be possible for a computer operator to handle measurements being called out for several soldiers at the same time. Also, the head fitter thought the measuring area was too congested to accommodate a computer. The computer table was set up at the back of the room where measurements were taken. Soldiers stopped at the table on their way to the clothing issue line. Each called out his name and measurements and they were entered into the computer.

There were two options for handling the size predictions. Option 1: when the soldier has been measured he receives a computer-printed sheet with size predictions and puts it on his clipboard. Fitters can consider the recommendations on the sheet when issuing garments. Option 2: the computer-printed sheet is held on file and just checked later against actual

sizes issued. Option 2 was selected because it caused less disruption of the normal processing (during the previous test option 2 was exercised successfully).

This test showed the expert system was practical (but not necessarily accurate enough) for the clothing issue facility environment. A single computer operator very nearly kept up with the workload on one of the busiest days. Two computers would be sufficient to handle all processing without delaying operations.

Speed of data entry could be improved by revising the expert system such that data is input in the same order as it appears on the clothing form. Then soldiers could simply sequentially read the measurements without referring back and forth to the computer screen and the clothing form. Also, operation of the system would be faster if it were set up for a rapid data-entry/print cycle. As it is, the operator has to accomplish several selections with the mouse in between processing of each soldier.

During the operational tests, performance of the expert system was not as good as human fitters. The case-based foundation of the system is sound, but the potential accuracy is limited by the validity of the cases. Several factors contribute to shortcomings in the cases used in the operational tests.

1. Measurements are not always accurate. Sometimes detail soldiers with little training take the measurements. Professional measurers make mistakes because they operate under time pressure and get tired from handling several hundred soldiers in a few hours.
2. Measurements are not consistent. Data in the cases was produced by different professionals and detail soldiers who do not necessarily employ the same measurement approach.
3. The measurements in the cases are not a sufficient basis to determine garment sizes. Human fitters can take other factors into account when selecting sizes for try-on, but the expert system is limited to the data in the cases. The cases themselves suggest the insufficiency of the measurements. There are situations where a group of soldiers all have almost the same body measurements, but several different sizes of a garment were issued to soldiers in that group. Access to additional data provided by a 3D body scan is expected to help this problem.
4. Variance in garment tolerances could cause inaccuracies. This variance can cause problems whether the size prediction is manual or automated. Given the assumption that out-of-tolerance garments are exceptional, the cases stored in the expert system should be based only on issues of in-tolerance garments. Therefore garment tolerances should be manually checked when cases are gathered to calibrate the expert system, to insure data in the cases is valid.

5. Certain garment designs were changed after cases were gathered for the expert system. According to the staff of the clothing issue facility at Fort Jackson, the new long sleeve shirt has a larger chest area than the old one. Also, long sleeve shirt sleeve length designations were changed. Since the new all weather black coat has a thinner liner than the old one, in some cases a smaller size can be issued. The new green coat seems to have a different fit. To account for such changes, the expert system could periodically be re-calibrated with a new set of cases using the new garments.

Further details about the system are provided in a report by Sarat Vemuri in Appendix A.

Size prediction system learning curve studies

Generally, performance of a case-based-reasoning system improves as the number of cases is increased, because for any new problem the chances increase of retrieving a case which is very similar. But at some point the rate of improvement will level off. Certainly cases should be added as long as they tend to improve system performance, but when the marginal benefit of adding cases becomes small, continuing to add them could be a wasted effort and could adversely affect system speed. Dr. Davis worked with volunteer graduate students who conducted three separate studies of system learning. The first study was done by Raktim Sen.

First Learning Curve Study

Sen's approach to gaining insight on this problem was to select randomly a set of test cases from the database of cases, and then measure system performance as cases were added in increments of 200. This process was tedious, because each increment of cases involved importing cases and indexing of the cases, which can take about an hour. Sen also helped determine how to test system performance. Sen's results were not very significant, but he did help determine that a better means of testing was necessary.

Detailed evaluation of the results of the Remind® built-in testing routines revealed that they are not satisfactory for system evaluation. The built-in routines allow random selection of a percentage of cases as "test cases" whose outcomes are assumed to be correct. Then the system automatically selects the test cases one by one and for each will retrieve a group containing a minimum of "n" most similar cases (similar in terms of body measurements) from the database (where "n" is a number selected by the operator). A report provides a summary of the similarity of the outcomes of the retrieved cases to the outcome of the test case (in this project, outcomes are garment sizes).

A significant shortcoming of the built-in testing routines is that they do not directly evaluate system predictions; in fact they do not even handle the concept of a system prediction. The ultimate purpose of this project is to produce a system which predicts (selects) garment sizes for an individual. To evaluate system performance for a prediction of a particular garment size, say the short sleeve shirt, one would like to analyze the accuracy of predictions. For example, for a group of 180 test cases one would like to know how often the system prediction is correct, how often it is off by a half size, a full size, etc. The desired report would look like the following:

Short Sleeve Shirt

Test Cases Number	Percent	Difference Between System Prediction and Test Case Outcome
100	55.6	0.0
60	33.3	0.5
10	5.6	1.0
10	5.6	1.5

The system may be designed to provide an ordered set of predictions, e.g. 1st choice size 15, 2d choice size 15.5, 3d choice 16.0. For this version of the system the test report should take into account the success of the alternate predictions, perhaps resulting in a report like the following:

Short Sleeve Shirt

Test Cases Number	Percent	Difference Between System Prediction and Test Case Outcome
100	55.6	0.0
60	33.3	0.5 (42 correct on 2d choice, 13 on 3d)
10	5.6	1.0 (6 correct on 2d choice, 2 on 3d)
10	5.6	1.5 (2 correct on 2d choice, 3 on 3d)

Second Learning Curve Study

The next study was performed by Dan Elias and Peter Kartawidjaja. Their work was more detailed than Raktim Sen's. Elias and Kartawidjaja developed "clusters" (search trees) for the case-based reasoning system for several garments. They then studied the "learning curve" of the system, to help determine how many cases are necessary for satisfactory results. The basic approach was to test the performance of the system for various numbers of stored cases. Ultimately one could use the resulting data to produce a graph of the learning curve (performance vs. number of stored cases). It was expected that this curve would rise rather rapidly at first, but then begin to level off at some point. Probably there would be no significant advantage to

be gained by adding more cases beyond that point. It was expected to be useful to know where the curve levels off to determine whether enough cases are on hand to test the current system adequately. Also, this information would help determine how many cases should be gathered when the 3D body scanner is available.

A batch testing program was made operational and was successfully transferred to the computer owned by one of the students working on this part of the project. This program allows running numerous tests with a single command. The students randomly selected 100 cases which would be used as test cases throughout this phase of the project. They tested the batch testing program on several sample files and confirmed that it worked properly and provided the output necessary to determine the learning curve.

When the study of the "learning curve" of the size prediction system was completed, Elias and Kartawidjaja decided to test the performance of the system for various numbers of stored cases by using the short sleeved shirt. The steps in the study included selecting a set of test cases, forming databases for various numbers of cases, converting each database to a CBR library file, testing each library file, and reporting results.

Selection of Test Cases

There are 4058 cases available from data collected at Fort Jackson, SC. Two hundred cases were selected randomly as test cases, using the RAND function in Lotus 1-2-3. These were stored in a file and were eliminated from the master database of cases.

Forming Databases

Two different approaches were employed in forming a set of databases of various sizes: random and stacked. The random approach involved creating a database "from scratch" by randomly selecting a certain number of cases from the master database. The stacked approach involved starting with a database of 20 randomly-selected cases, and then building larger databases by progressively adding more cases from the master database. Using each method, databases of the following sizes were generated: 20, 40, 60, 80, 100, 150, 200, 300, 400, 500, 1000, 2000, 3000, and 3858.

Converting to CBR Library

Initially a new library with an appropriate format had to be created, to allow importing data from the databases. The "outcome" field and all "match" field types were set to "REAL" so that the library would be compatible with the testing program RETR1.EXE which had been created

earlier. Short-sleeve-shirt-size was the outcome and the match fields for the inductive search were chest, height, neck, waist, and weight. The nearest neighbor search weight vectors were:

chest	16	(33%)
height	8	(17%)
neck	8	(17%)
waist	8	(17%)
weight	8	(17%)

The first step was to activate the new library in Remind®, and then import from one of the databases. Those cases were designated as "stored". The test cases were imported, and were left with the hypothetical designation. Finally, the new library was clustered, meaning Remind® automatically built an index tree which used the features of the stored cases to account for differences in outcome field values (short sleeve shirt sizes).

Testing

Once a library was formed, it was tested using the RETR1.EXE testing program which automatically identifies the test (hypothetical) cases in the library, and for each conducts a test and stores the results in a file. For each test case, RETR.EXE performed an inductive retrieval followed by a nearest neighbor search. Finally, voting was employed for the 10 cases which had the highest similarity score assigned by the nearest neighbor search. The size receiving the highest number of votes was designated as first choice, and so on for the second and third choice. Results of each test case appear in one row of the output file, and include the three chosen sizes, the number of votes each received, and the "correct" size. The "correct" size is taken from the test case; the assumption is that it was correctly assigned by a human fitter. The testing program counts the number of times the first, second, and third choices were correct, and the number of times none of the choices were correct.

Refining the Set of Test Cases

Before attempting to test system performance, the test set was purged of cases which seemed to be erroneous. Sources of errors include rounding of measurements, mistakes in measurement, mistakes in recording data on the clothing form, non-standard tolerances on garments, and mislabeled garments. To identify problematic cases, the first test was conducted with the largest available library of 3858 cases. A case was purged if among the 10 retrieved cases there were 8 or more votes for a different size. Nineteen test cases were purged, leaving 181.

Results of the Second Learning Curve Study

The study produced a significant amount of data. The main findings are concisely reported here. Because of the way cases were randomly selected from scratch in the random method, system performance was rather erratic as the number of stored cases increased. But with the stacked method, system performance tended to improve as the number of stored cases increased, as would be expected. However, there were a few situations where the performance became worse as more cases were added (Table 4).

Table 4. Results of Testing with the Stacked Approach

<u># Stored Cases</u>	<u>Percentage of Correct First Choice Predictions for 181 Test Cases</u>
20	55.2
40	62.4
60	64.2
80	67.4
100	69.0
150	64.4
200	67.9
300	71.8
400	70.3
500	71.3
1000	66.3
2000	71.8
3000	71.3
3858	70.2

Third Learning Curve Study

A volunteer graduate student, Vinit Jindal, performed research which contributed to this project, with Steve Davis advising him. His objective was to determine the "learning curve" of the revised expert system. The case-based system should perform better as more cases are added, but one would expect that at some point the rate of increase in performance should begin to level off. A study of this sort should help answer the question, "how many cases are enough?"

Jindal did the most extensive study. The main finding was that, as one would expect, learning increased relatively rapidly at first and then leveled off as more cases were added to the database. But it was surprising to find that even with just a few cases in the database, predictions were nearly 50% correct. For the short sleeve shirt, when cases were increased to 4000 the performance only increased to 67%. This indicates that, for this database, the learning curve is rather shallow.

Jindal's report (see Appendix B) summarizes the results of the study of learning with short sleeve shirt and trousers. For various numbers of cases in the library it was determined what percentage of predictions were correct for 100 randomly selected cases. Results are shown for the first, second, and third choice of system predictions.

3D Body Scanner Interface Software Development

The project team was divided into two groups. The expert system group focused on developing the size prediction software. The 3D group focused on developing methods of extracting measurements from a 3D body scan. The initial effort of the 3D group concentrated on developing demonstration programs on two computer systems: a SUN workstation and an IBM PS/2. The SUN demo runs under the X-Window System and allows the user to view and manipulate 3D objects on the screen. User options include automatic rotation, increase and decrease of speed of rotation, manual rotation in each of the three dimensions, and scaling of the object up and down. The user also has the option of selecting among several datafiles, quickly displaying one or more in individual windows.

In order to develop this demonstration, a graphical library was built for the 3D object display. The library contained routines that perform 3D manipulations. This includes 3D transformation, translation, scaling and rotation, and 3D to 2D projection. The library is independent of the hardware platform, and thus is easily ported to different machines, such as the IBM PS/2.

A demonstration of preliminary 3D software was developed for the Sun workstation. This software converts the x-, y-, z-coordinates in the file from the 3D scanner to a computer image of the person's torso on the screen. The software allows the user to rotate the image to virtually any perspective. The data diskette with the output from the 3D scanner was obtained from Jud Early of [TC]² during the meeting with him in Charlotte in mid-July 1992 (see page 5 and Figure 1, page 6).

A similar demonstration was developed for an IBM PC compatible machine. It was displayed at the Clemson Apparel Research booth at the 1992 Bobbin Show in Atlanta (September 15-18). The PC version displayed the rotating image of a torso. It also highlighted in color three cross sections (chest, waist, and seat) and displayed in a small box on the screen the circumferences of these body dimensions with accompanying predicted military uniform sizes. In the final system being developed it is expected that these measurements will be sent directly from the 3D scanner to the case-based (expert) system which will predict the size uniform that a recruit will initially try on.

The Bobbin Show demonstration program ran under Microsoft Windows version 3.1. Borland C++ and Object Windows were the chosen tools for the implementation.

The next step was to port the code running under X-Windows onto the PC. A framework was developed which supports drawing facilities, this being the device-dependent code. Next, the device-independent code for 3-D viewing operation was integrated into the framework. A few modifications were made to bypass the memory restrictions on the PC. For example, the large memory model had to be used.

Enhancements to the software were started. These included: (a) allowing the user to click on an arbitrary horizontal "slice" of the torso, not just the chest, waist, or seat, and to take its measurement; (b) allowing the user to click on two points on the body to take a surface measurement from a first to a second point, or through a second point to a third point; (c) selecting from any number of image files and viewing the image; (d) displaying multiple images on the screen simultaneously, or equivalently, multiple perspectives of the same image simultaneously; and (e) developing a library of functions to automate the measurements (i.e., to enable the software to determine the location of different parts of the body and then measure them).

Point-and-click Measurement, Circumferences and Straight-Line Distances

An initial version of "point-and-click" measurement was developed. The software was developed to allow a user with a mouse to select interactively and highlight a horizontal slice of the human figure or a line segment between two random points on the body surface (Figure 2). If a horizontal slice is selected, the software measures the length of the circumference of the slice. If two points are selected, the software takes the shortest surface distance between the two points. In either case, the measurement value is displayed in a window in the upper right hand corner of the screen.

The user continued to be able to modify his or her perspective of the figure, either translating the figure in the X-, Y-, or Z-direction, or rotating the figure in the X-, Y-, or Z-direction. If the figure is either translated or rotated, the selected slice or the line segment continues to be highlighted. This allows the user to view the slice or segment from virtually any direction. at this point the code ran on SUN workstations only.

Early in the project, very preliminary work was done on automatic identification of body dimension locations. This requires that the software correctly identify parts of the body, the chest for instance. If the data representing the body is complete, i.e., if an accurate scan has been obtained from the neck to mid-thigh, chest identification and measurement can be done easily. The software simply searches for the slice just below the armpits. Problems occur, however, when the data is incomplete. For example, if the data describes a figure with no arms or only a single arm, chest identification is difficult. It is assumed that an improved body scanner will eliminate this incomplete data problem.

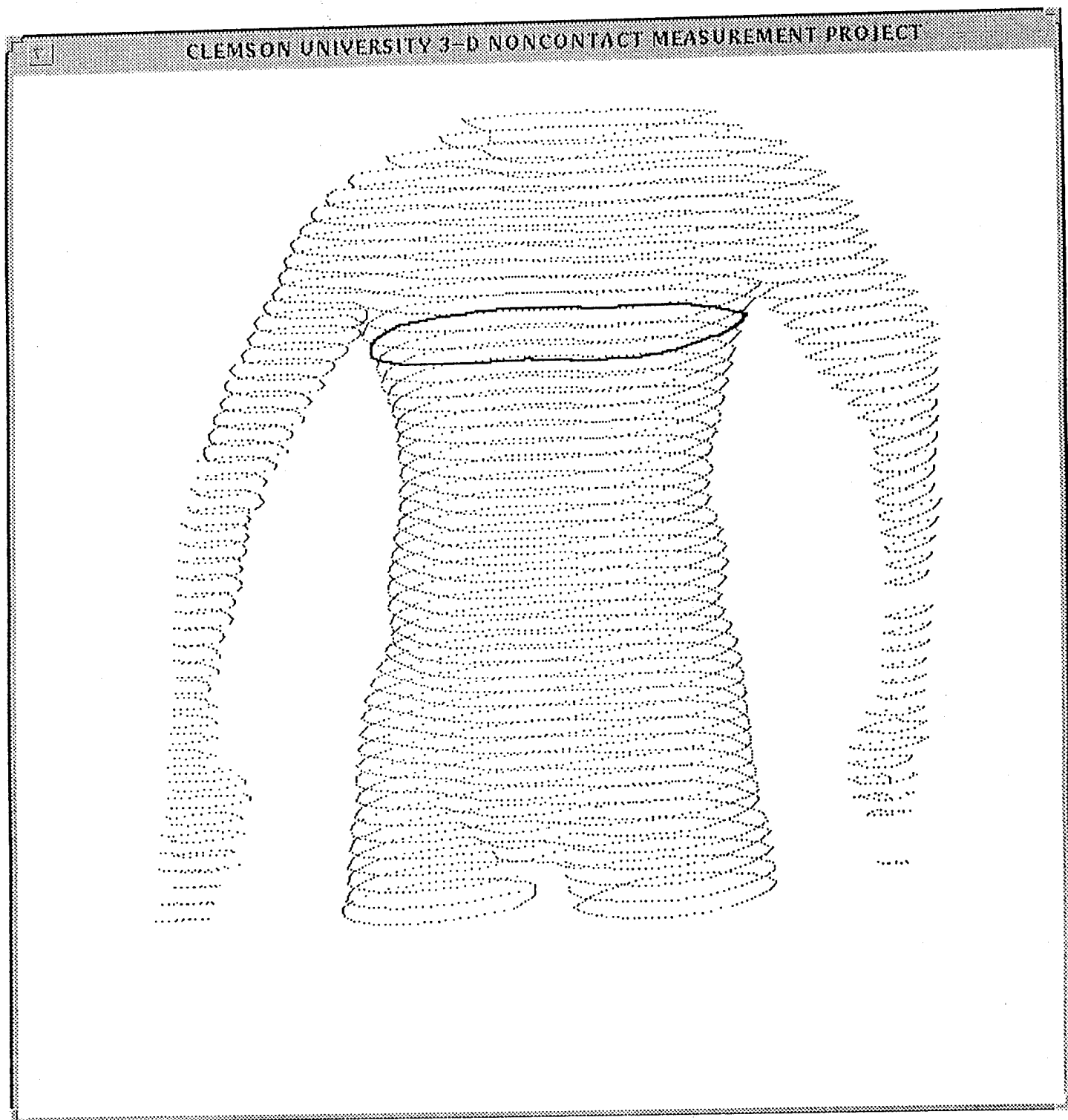


Figure 2. Highlighted slice for circumference measurement.

Porting Code from SUN Workstations to IBM PC-compatibles

Guidelines were developed for porting the software developed on a SUN workstation to an IBM PC-compatible running Microsoft Windows. These guidelines are listed below.

Generality

If a called function needs a structure from the calling function, the structure should be passed as an argument and should not be accessed as a global variable. This has the advantage that any person reading the code can understand (sometimes even without comments) what that function does and what kind of arguments it takes.

Since by nature C++ is a language with the philosophy that implementation details should be hidden, and since nearly all interaction takes place between classes, it becomes increasingly difficult to integrate code which makes use of global variables. Moreover, disallowing the use of global variables makes the code more general and easier to change.

More Emphasis on Commenting

All functions should be preceded with one or two lines describing what it does.

Modularity

Standard conventions should be followed regarding .h files. All .h files should contain the prototypes of all accessible functions in a certain module. .H files should contain information on the functions defined in the corresponding .c file. When it comes to porting, .h files would be needed for declaring external "C" functions so that these can be linked properly. NOTE: .H files contain particulars of the data (size, number of points) which .c files are programs.

The User Interface

The User Interface of the SUN version of the 3D code was separated from the remainder of the code in order to facilitate porting the software to the IBM-PC compatible running Microsoft Windows. The User Interface consists of all windowing-related functions.

More options were added to the user interface. The user could change the angle of rotation, translation offset, and scale factor using a potentiometer-like measuring device. In addition, automatic rotation about individual axes was made possible. Finally, color was added to all the windows and buttons in the menu window, making the menu buttons more realistic.

Point and Click Surface Measurement

The next code development was to take the surface distance measurement from one arbitrary point on the body to another. The method selected was Dijkstra's algorithm which can be used to compute the shortest path between points on an undirected graph.

In order to use Dijkstra's algorithm, it was necessary to impose a graph on the set of discrete points. Recall that all that is available is a set of points expressed in X, Y, Z coordinates in 3-D space. Because the points are laid out in rows and columns on the person's body, a natural approach seemed to be to connect points to their nearest neighbors. For example, if each point were connected to its north, east, west, and south neighbors, then the following interconnection pattern emerges:

```

      |   |   |   |   |
x --- x --- x --- x --- x --- x ---
      |   |   |   |   |
x --- x --- x --- x --- x --- x ---
      |   |   |   |   |
x --- x --- x --- x --- x --- x ---
      |   |   |   |   |

```

Dijkstra's algorithm, with this squarish pattern, did not produce smooth paths between source and destination points. A period of experimentation with other patterns followed. The objective was to find an interconnection which produced reasonably smooth paths from source to destination nodes. The pattern that provided the desired smoothness is described below. Let the x's below represent points on the 3D image. Consider the point "X" in the center.

```

-5  -4  -3  -2  -1  0  1  2  3  4  5
x   x   x   x   x   x   x   x   x   x   x

           x   X   x

x   x   x   x   x   x   x   x   x   x   x
-5  -4  -3  -2  -1  0  1  2  3  4  5

```

Number the points in the rows above and below point "X" as shown above. The point directly above and below X are numbered 0. Points to the right of points 0 are numbered 1, 2, ... Points to the left of points 0 are numbered -1, -2, ... An edge is defined between point X and points -5, -3, -2, -1, 0, 1, 2, 3, and 5 in the rows above and below X. In addition, edges are formed between X and points immediately to its left and to its right.

Defining more edges will lead, at least in the short term, to smoother paths, but at the expense of longer computation. A software design decision was later made to define which graph to impose on the set of points. A possible option could be to allow the user to select among several graphs, selecting between path smoothness and speed of computation.

After the graph is defined, it is possible to apply Dijkstra's algorithm. The algorithm is called a breadth-first search for the destination point starting from the source point. As the search widens, more points are examined and compared with the destination point. The search approach is very much like that of an ever-widening ripple formed when one tosses a pebble on a still lake. Once a destination point is discovered, the path that led to the point is determined and returned by the algorithm. The interested reader may learn more about Dijkstra's algorithm in a book on computer data structures [for example, Weiss, M. (1992). *Data structures and algorithm analysis*. Redwood City, CA: Benjamin Cummings]. This basic algorithm allows a variety of surface measurements to be taken on the 3D image.

Single Source, Multiple Destination (Radial) Measurement

In single source multiple destination (radial) measuring the objective is to define first the source point (X1), and then a set of destination points (X2, X3, ..., XN). The strategy is to apply Dijkstra's algorithm starting with the source point and spreading out in a breadth-first fashion, as described above. As destination points are discovered, the distances from the source point are recorded. Dijkstra's algorithm continues to be applied, however, for as long as undiscovered points remain in the destination set. It defines the set of (n-1) measurements from X1 to X2, from X1 to X3, ..., from X1 to Xn. We start at X1 and conduct a search outward collecting paths first to the nearest points, and then to points further and further away. At each step, we check to see if the new points examined contain any of the points X2 through Xn. If the point Xk ($2 \leq k \leq n$) is found among the most recently examined points, the path from X1 to Xk is determined, the edges are highlighted, the sum of the edges is taken, and the outward search for points continues. The process terminates when the destination point most distant from the source point is discovered. The (n-1) measurements are displayed in the MEASUREMENTS window on the computer screen.

This strategy provides results quickly because all that is required is a single application of Dijkstra's algorithm. At each step, the set of destination points is compared with the points most recently covered by Dijkstra's algorithm. If any destination points are found, their distances from the source point are recorded and the destination points are marked as discovered.

Multiple Point Measurement Along the Body Contour

In multiple point measurement along the body contour, the first objective is to define a sequence of points, $x_1, x_2, \dots, x_N$ and measure the distances between them $(x_1, x_2), (x_2, x_3), \dots, (x_{N-1}, x_N)$. The strategy is to apply Dijkstra's algorithm $N-1$ times. In the first application, x_1 is the source point and x_2 is the destination point. In the second application, x_2 is the source and x_3 is the destination, and so on. This operation requires multiple calls to Dijkstra's algorithm and takes the corresponding amount of time to complete.

This operation is a perfect application for parallel processing. If multiple processors were available, it would be possible for each processor to take a distinct measurement independently. The operation could then proceed P times faster where P is the number of processors available. Parallel processing is a technology that has become cost-effective for PC's and may be considered in the future.

After allowing the user to click on several points on the body: $X_1, X_2, X_3, \dots, X_n$, multipoint measurement takes the sum of surface distances between X_1 and X_2 , between X_2 and X_3 , ..., between X_{n-1} and X_n . For example, multipoint measurement may be used to measure the distance between a point on the person's center-back, to the shoulder, to the elbow, to the wrist (for sleeve length definition). The path on the body is displayed as the measurements are taken (Figure 3). This allows the user to verify visually that the measurement taken is accurate. The Dijkstra's algorithm graph selected for surface measurement connects each point to twenty neighboring points. This gives a smooth appearance to the paths formed when taking multipoint and radial measurements.

Development of PC software

After creating the new tools on the Sun, the focus of the 3D software development was on improving the PC version. The execution of multipoint measurement was particularly slow, taking approximately 2.5 minutes to take a measurement from center-back to shoulder to elbow to wrist. This was considered totally unacceptable execution time.

The first improvement was to represent the data in more compact form, specifically reducing the space required for the representation of a numerical value from 4 bytes down to 2 bytes. This reduction in space, allowed the entire set of over 8,000 body points and the induced graph which interconnects the points (consisting of approximately $18 \times 8,000 = 144,000$ edges) to be loaded into memory at one time. Being able to access all values immediately, because they are all contained in memory, made a significant contribution to the reduction of execution speed.

Searching for neighboring points also became more efficient. The path from a point to any other point is formed by examining each edge from the source to

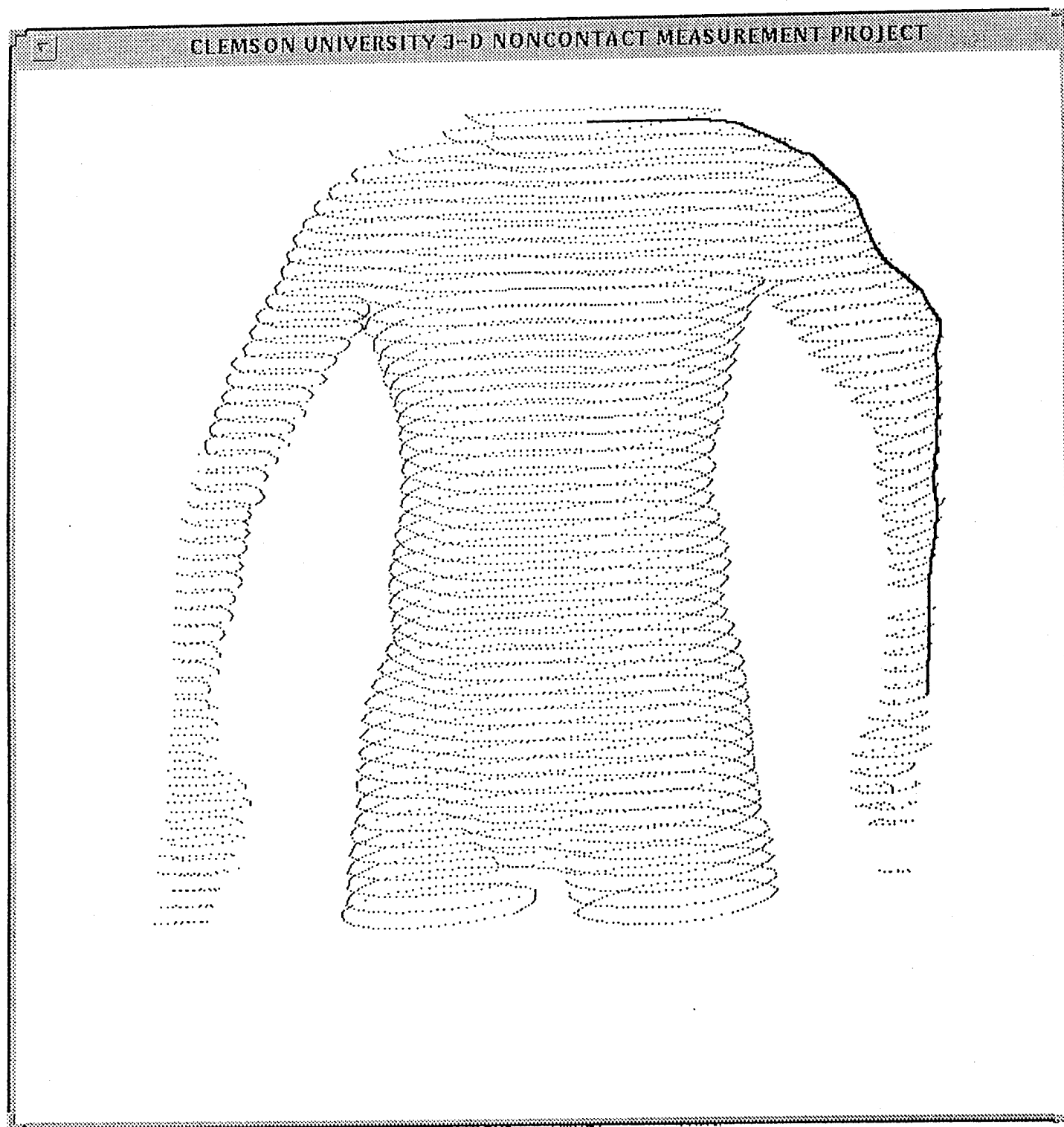


Figure 3. Highlighted surface distance to be measured (before smoothing).

its neighboring points, and then edges from the neighboring points to their neighboring points, and so on, until the destination point is discovered. The basic step is therefore to discover quickly which points are neighbors to a point. The software performing this basic step was improved and also contributed to the improvement in execution speed.

Finally, the code was ported to a faster PC, specifically a PC with a 33 megahertz Intel 80486 processor running Windows 3.1. These three improvements to the PC version of the code reduced execution time by approximately 80%. The time to measure from center-back to shoulder to elbow to wrist was reduced to approximately 30 seconds. Measurements of 30 seconds or less, should be acceptable to the end user.

Continued Tool Development—Vertical Slices

The 3D software development group created tools to describe vertical "slices" of the body image. If we describe the body as existing in 3D space, i.e., X-, Y-, and Z-space, a slice of the body is defined simply as the subset of points of the body image obtained by setting one of the dimensions to a constant value (in the circumferences already defined, the slice had a constant Y). If we fix the X-value to some constant and allow the Y- and Z-values to vary, we obtain a vertical slice of the body from front to back. If we allow the X-value to increase or decrease, we get different vertical slices of the body. Similarly, if we fix the Z-value to a constant and allow the X- and Y-values to vary, we obtain a vertical slice of the body, this time from the left to the right side. These slices can provide information regarding the posture of individuals, thus allowing for more accurate fitting of shirts and jackets in special measurements uniform issue (Figures 4 and 5).

The user may elect to take slices along the X- or along the Z-axis. For example, the software allows the user to select a point x_0 on the X-axis using the cursor arrows as necessary. A new window displays points of the body on or near the vertical plane $X=x_0$. If the body is facing front, the result is a side view of a front-to-back slice of the body, containing parts of the chest and the back.

The user may also select a point z_0 along the Z-axis and generate a display of points of the body on or near the vertical plane $Z=z_0$. If the body is facing front, the result is a frontal view of a slice of the body extending from one arm to the other.

The slices provide selected views of the body allowing the user to focus on specific parts of the body. For example, a slice along the X-axis may show the curvature of the person's body along a line down the center of his or her back. The same slice would give a profile of the person's waist or seat. Similarly, a slice along the Z-axis can isolate the person's shoulders and highlight their slope.

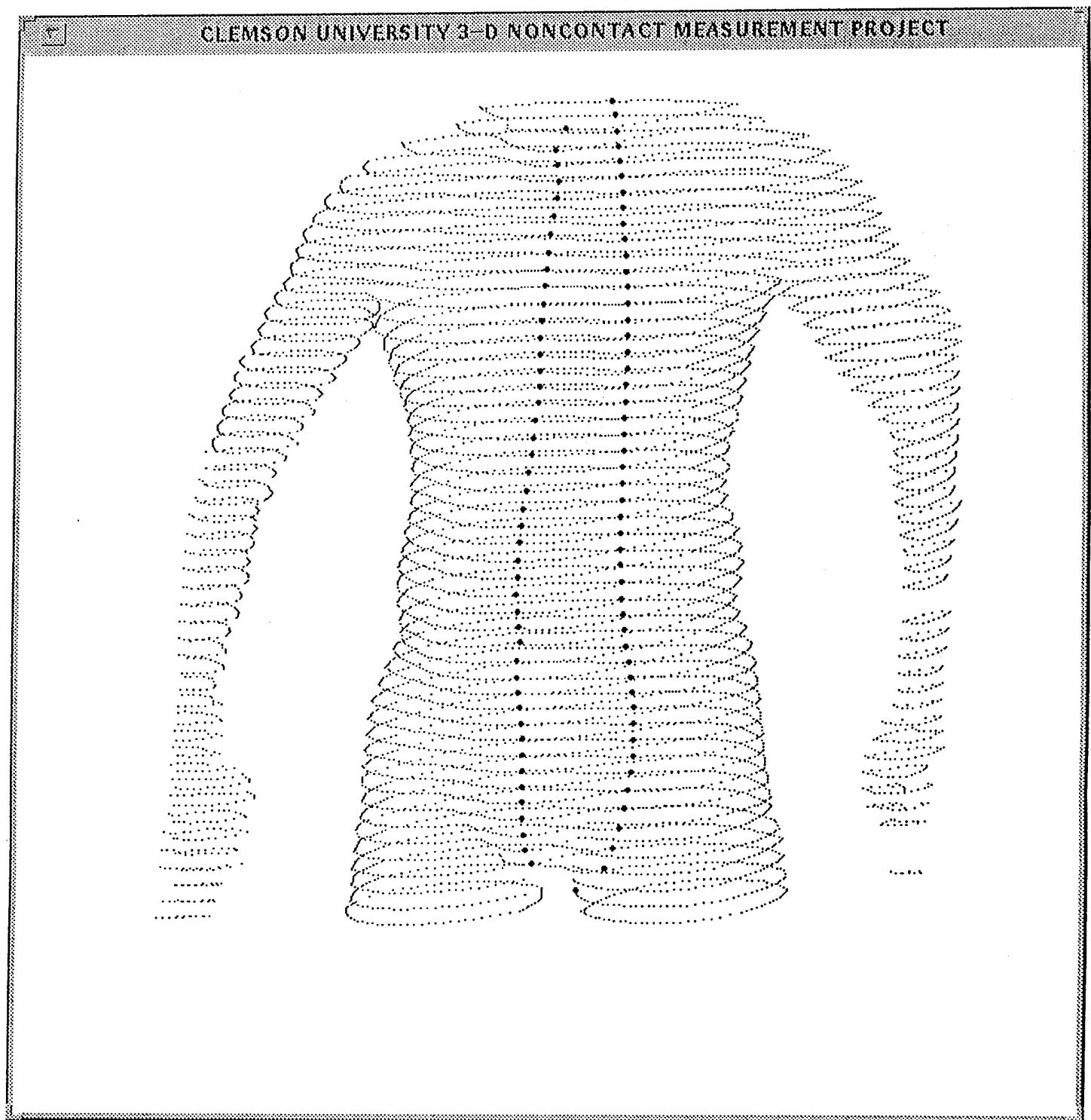


Figure 4. Vertical slice selected.

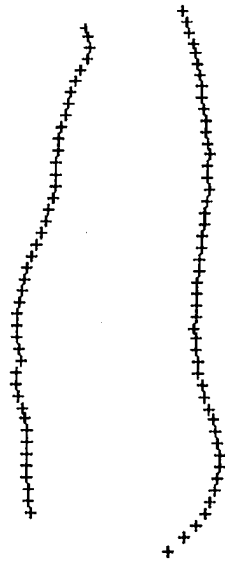


Figure 5. Vertical slice displayed in side view.

The next step in this development was to allow the user to take measurements along the selected slice. The first measurement considered allows the user to click on a source point and a destination point. The software takes a measurement from the source to the destination points along the surface of the body. This allows, for example, a measurement from the base of the neck to the waist of a person along the person's spine.

Automatic Body Part Recognition

Following the project team review and planning meeting in July 1993, for the next several months, the 3D software development effort focused on automatic body part recognition. This software, when complete, could determine, for example, the chest, waist, seat, sleeve length, etc. of the scanned body without user intervention. The primary objective was to have the computer detect specific body parts and measure each automatically.

The first step in automatic recognition of body parts was a general partitioning of the body into its basic components. The project team chose to construct a hierarchical partitioning of the body. At the highest level, Level 1, the components are the person's head and neck, torso, arms and legs. Each of the components can then be partitioned further. For example, the torso can be further partitioned into the Level 2 components: the seat, the waist, and the chest. Similarly, an arm can be further partitioned into Level 2 components: upper arm, lower arm, wrist and hand, etc. How many levels are used, i.e., how much information is needed, determines how detailed the partitioning should be.

The data used is organized in such a way that a horizontal "slice" of the body is the basic unit. The arms, legs, and torso are each divided into slices. The enumeration of the slices proceeds from the neck downward. This implies two things. First, a body part will simply be a collection of data "slices". So, for example, the left arm may be composed of slices {23, 26, 29, 32, ...}. And second, individual slice information must be used to determine which body part a slice is a member of. Two such pieces of information are: (a) slice circumference and (b) slice position.

The initial approach use identified Level 1 components as follows:

1. Recognize arms and legs using both slice position and slice circumference. The remaining slices comprise the torso.
2. Identify the arm pits again using slice circumference.
3. Identify the chest line on the torso. This can be done by identifying the first slice under the arm pit.
4. Recognize the chest area, i.e., the area about the chest line.
5. The remaining slices in the torso comprise the lower torso.
6. Divide the lower torso into the waist area and the seat area.

7. Find the fullest slice, i.e. the slice whose distance between the front and the rear is the largest and near the bottom of the lower torso. This we call the Seat.
8. Identify the waist as that slice near the top of the lower torso which either has the smallest circumference if the circumferences near the top of the lower torso are decreasing, or has the largest circumference if the circumferences near the top are increasing.

The programming primitives used to determine these body component parts are building blocks for automatic measurement definition. This was accomplished through a command interpreter. The general format of a command is:

command *option*

where a command is one of four types: Region, Slice, Point, and Measurement (options vary with each command).

A Region command is used to select a region of the body. The user may then enter a Slice command to select a specific slice within the region. Point commands allow the user to select a point within a slice. Slices and points may be saved and used with other Slice, Point, and Measurement commands. Subcommands within each command type are described in detail below.

Commands are *not* case sensitive, thus **REGION TORSO** and **region torso** are equivalent. Commands and options must be spelled out in full. The messages "Command not recognized" or "Option not recognized" are generated whenever commands or options are misspelled.

Region Command

The format of the **region** command is **region area** where *area* is one of **TORSO**, **LARM**, **RARM**, **LLEG**, **RLEG**, **UPBODY**, **ALL**. This command selects the part of the body that the user wants to study further and measure. The areas, respectively, are the torso, left arm, right arm, left leg, right leg, upper body, and the entire body. An area must be selected before the user may issue Slice commands. If a slice command is issued before an area is selected, an error message will be generated.

Slice Commands

There are several Slice commands available.

selectslice percent where *percent* is an integer between 0 and 100. This command allows the user to specify a slice some percentage distance from the top of the region, where the distance from top to bottom represents 100%. **Selectslice 10**, for example, will select the slice approximately 10 percent from the top of the region. For convenience, three often used percentages are fixed, namely **topslice** (which selects

the top slice of the selected region and is equivalent to **selectslice 0**), **middleslice** (which selects the middle slice of the selected region and is equivalent to **selectslice 50**), and **bottomslice** (which selects the bottom slice of the selected region and is equivalent to **selectslice 100**).

slicemove *direction distance* where *direction* is either *UP* or *DOWN* and *distance* is a positive numeric value representing a distance in inches. For example, **slicemove UP 1.5** moves up from the current slice a distance of 1.5 inches. The newly selected slice becomes the current slice. The current slice may be saved for future use.

slicesave *name* saves the current slice and assigns to it a *name*. This *name* may be used in other Slice commands (see **maxslice** and **minslic** below). *Name* may be any string, starting with a non-blank character and continuing with any character (including blanks and punctuation) up to a maximum length of twenty characters. The following are all valid names: *mid-thigh*, *abdominal mid-point*, *slice 12*, *KNEE CAP*, +++++. Names must be used with the following two commands.

maxslice *slice1 slice2* where *slice1* and *slice2* are slice names (see the **slicesave** command above). The command **maxslice** selects from the current region the slice with the largest circumference between *slice1* and *slice2*. *Slice1* must be above *slice2*. In the case of a tie, the slice closest to *slice1* is selected.

minslic *slice1 slice2* where *slice1* and *slice2* are slice names (see the **slicesave** command above). The command **minslic** selects from the current region the slice with the smallest circumference between *slice1* and *slice2*. *Slice1* must be above *slice2*. In the case of a tie, the slice closest to *slice1* is selected.

Point Commands

Point commands allow a user to select individual points within a slice. Two commands available are the **center** and **most** commands.

center *position* where *position* is one of *FRONT*, *BACK*, *LEFT*, *RIGHT*. This command selects the point closest to the *center front*, *center back*, *center left*, *center right* of the current slice.

most *position* where *position* is one of *FRONT*, *BACK*, *LEFT*, *RIGHT*. This command selects the *front most*, *back most*, *left most*, or *right most* point of the current slice.

Further development work proceeded on measurement commands and control commands. Eventually these building blocks of commands can be combined into macros which will select landmarks, measure the 3D scan, and record measurements in a data file to be sent to the size prediction expert

system. An interactive step requiring approval of the selected measurement sites by the operator will be included.

The next two commands included in the command interpreter were: (a) **surfacedist** and (b) **lineardist**, which compute the surface and linear distances between two selected points, respectively. The user must first select two points. This may be performed by a sequence of **selectregion**, **selectslice**, **slicemove**, **selectpoint**, **pointmove**, and **pointsave** commands.

Assume that the user has saved two points and has named them P1 and P2. The user may then issue a

surfacedist P1 P2

command naming the points. This will call a function which determines the shortest path from P1 to P2 along the surface of the body points. The process produces a path and the distance from P1 to P2, closely approximating measurements that would be taken using a tape measure on a real person. "**Lineardist P1 P2**" performs a similar function. The difference is that **lineardist** takes the direct distance between P1 and P2, without measuring along the surface of the body.

It should be noted that **surfacedist** and **lineardist**, as well as all of the other commands (**selectregion**, **selectslice**, etc.) are instructions which can be interpreted by the computer. This means that the user can write programs which the computer can interpret and execute without the user having to point and click a mouse. In brief, the user will eventually be able to enter commands such as "chest" and the computer will automatically find and measure the chest, or "waist" and the computer will automatically find and measure the waist, or "sleeve" and the computer will automatically find and measure the surface distance from the center-back, through the shoulder, through the elbow, to the wrist. The ultimate in convenience will be when the user is able to issue a command "all" and the computer takes all the different programmed measurements the user wants to see.

A complete description of these functions is located in Appendix C. A copy of the source code for 3DM is located in Appendix D.

Tracking 3D body scanning technology

On December 22, 1992 Dr. Nancy Staples visited DMS in East Rutherford, NJ to see the scanning equipment as revised through [TC]² and to discuss the progress in the development of the software to run the scanner. Dr. Staples emphasized that the CAR project team needs only the scanning capability and the X, Y, Z output with no additional software. Apparently there are others who were requiring more software development before the equipment was ready for their use. Peter Kuhlman was not able to project a target date

for completion of the camera-merging software, the remaining development needed before the equipment would be ready for CAR.

The project team continued to be concerned about the availability of the Dimensional Measurement Systems 3D Body Scanner. The original two-camera system with ten-percent interpolation in each 180° would have been sufficient for this project's needs. Due to the infusion of funds into DMS for the development of a full-body, minimal interpolation six-camera system, the two-camera system was dismantled and parts robbed from it. The six-camera prototype was not complete and there were software problems which have yet to be solved. DMS sent sample files to the CAR team to enlist our help in solving the problems, but the data were incomplete. As a result, the project team began again investigating alternative vendors for body scanning.

On March 12, 1993 Dr. Nancy Staples and Dr. Roy Pargas made a trip to Monterey, California to visit Cyberware, Inc. The purpose of the visit was to observe the 3D scanners being developed at the company. Cyberware did not yet have a 3D full-body scanner but, but had a working prototype and was confident that they had the technology to build one. At that time, the company had 10 years experience developing smaller scanners, such as head scanners being used by the U.S. Air Force to do research on helmets and leg scanners being used by medical physicians to design and build artificial limbs, specifically legs. Cyberware appeared to be a possible candidate for a supplier of a 3D full-body scanner. On April 1 and 2, Steve Addleman, Vice President of Cyberware, visited Clemson Apparel Research to see our operation and to discuss further the possibility of our collaboration.

In early May 1993 the remains of DMS, now defunct, were shipped to Raleigh, NC where [TC]² took over the development of the product. The two programmers previously employed by DMS in NJ moved to NC. [TC]² engineers are still trying to fix the hardware problems while the programmers try to fix the software problems. This equipment was not ready in time for the CAR 3D project to use in a field test.

Because of the status of the former DMS scanner, investigation continued for alternatives (including other equipment vendors and funds for lease or purchase). The project team submitted a proposal for equipment purchase to the Defense Experimental Program to Stimulate Competitive Research. Notification of award was expected by June 30, 1993. The request was denied.

On May 17, 1993 Lon Crosby and Guy Grotke of Numedloc, developers of a 3D body scanner employing biostereometrics, visited Clemson Apparel Research to discuss the viability of their product for the 3D project. Biostereometrics employs a technique similar to what is used in cartography to determine the x-, y-, z-location of points defining the contour of the body. Numedloc was going to produce a prototype to the CAR team's specifications and demonstrate it in mid July. The date was postponed until mid August, then September. At that time it was to be determined whether their

equipment would be useful within a few weeks or if it would be at least three months before they had a field-test-worthy product. A preliminary file was received, which indicated that they were not as far along as they had implied. If the data had looked sufficient for the project's needs, Dr. Staples and Dr. Pargas were going to visit the laboratory and if the equipment had been approved a system would have been sent to Clemson Apparel Research for experimentation and debugging before installation at Fort Jackson. Nothing further was heard from Numedloc after September 1993.

The search for a suitable scanner continued. No company had yet been able to provide a full-body scanner capable of generating the data required by this project and at a cost less than \$170 thousand. Discussions continued, however, between the principal investigators of this project and various scanner development groups. It was deemed possible to create a consortium of interested parties who would pool their financial resources in order to fund a prototype from a company with a known history of product development and delivery.

Preparations were made for an early February 1994 meeting to allow two potential scanning device companies to present their products and ideas to Natick anthropologists, DLA's Julie Tsao, and representatives of two major apparel manufacturing corporations. The meeting was to include the vendor presentations and a demonstration of the Clemson software for extracting body measurements.

On February 4 a group was gathered at Clemson Apparel Research for the purpose of introducing 3D, non-contact body scanning to potential users of the technology. It was originally intended that a two-day meeting would be held, with Cyberware of Monterey, CA presenting on February 3 and LaserDesign of Minneapolis, MN presenting on February 4. However, Cyberware dropped out in late January and the agenda was consolidated on February 4. Participants included representatives of two major U.S. apparel manufacturers, U.S. Army Natick physical anthropologists, and the principal investigators of the current CAR/DLA project. The DLA was not represented.

At the conclusion of the day the groundwork had been laid for forming a consortium whereby the potential users would contribute funds to a project in which each participant would receive a scanner from LaserDesign and software from Clemson. A proposal was drafted for review in late April, followed by presentations to be made to corporate decision makers in May. Upon commitment of funding, each participant would receive a scanner from LaserDesign in approximately six months (preceded by the alpha testing of the initial prototype by Clemson) and two years access to Clemson software development (to customize the usefulness of the output to the participant). It was hoped that the existence of this consortium would provide the impetus for this project to acquire a scanner for completion of the project's work.

A proposal was developed with the CAR team as subcontractor for providing a measurement extraction module for LaserDesign's DataSculpt product.

The proposal and a video tape with computer graphics of the proposed scanner were sent by LaserDesign to the apparel manufacturing participants in the February meeting at CAR as well as to potential consortium members in apparel retail, health care, and fitness. Visits were made to interested companies in late April with the goal of acquiring funds to speed the availability of a field-test-worthy scanner. On April 18 a presentation was given to a gathering at VF Corporation Information Technologies Services in Greensboro, NC. On April 20 a demonstration was given to product development and management representatives of Russell Corporation in Alexander City, AL. The software for extracting measurements from a 3D, non-contact body scan was demonstrated by Dr. Pargas and Dr. Staples and information about the Laser Design scanner was presented by Marty Schuster. A third company cancelled their appointment a week before their scheduled date.

Unfortunately the enthusiasm of February did not carry over to April. None of the companies approached was willing to commit funds until the development work was completed.

At the close of this DLA project, Cyberware of Monterey, CA and Laser Design of Minneapolis, MN appear to be the top contenders for producing the first useful full-body scanners in the United States. A Cyberware newsletter states that demonstrations will be given in Monterey, CA in fall 1994 with deliveries to begin in early 1995. Laser Design plans to have a prototype ready by late September 1994 with deliveries to begin three months later.

Fact Finding Visit to Wright Patterson Air Force Base

On April 13, 1994 Dr. Pargas and Dr. Staples visited the Armstrong Center at Wright Patterson Air Force Base. Discussions were held with Kathy Robinette and her staff concerning their work in headgear development using Cyberware scanning. A video tape of the CAR measurement extraction software was shown to the Armstrong Center staff and to a group of visitors from the University of Surrey, UK (developers of a laser-based head scanning system in the UK). Kathy Robinette suggested that Dr. Pargas and Dr. Staples visit in England with the Surrey group, with Peter Jones at the University of Loughborough (developer of Loughborough Apparel Scanning System—LASS), and with Marks and Spencer (retailer with whom Peter Jones collaborated on a scanning project). Dennis Burnsides, contractor with the Air Force, employed by Systronics, Inc. was most interested in using Clemson's method for finding the shortest path of a surface distance on a body scan. CAR and the Armstrong Center will continue to keep each other informed of progress in their respective projects.

Investigation of British Scanning Devices

Initial plans were made to investigate scanning devices and software in Great Britain. The purpose of this investigation was twofold: to gather information

regarding the current state of research efforts in the area of human, full-body, 3D, non-contact scanning in Great Britain and to determine whether any of the scanning devices there could use Clemson's measurement extraction software. This effort had been encouraged by both Kathleen Robinette at Wright Patterson Air Force Base and Steve Pacquette at U.S. Army Natick.

Through Robinette, Pacquette, and letters of inquiry to the CAR team from researchers in Great Britain, it was determined that three groups would be worth investigating. The National Engineering Laboratory in Glasgow was developing a moire-fringe topography-based system for the Defence Clothing and Textiles Authority. Also involved in this project was the Stores and Clothing Research and Development Establishment. A second research effort was being conducted by the University of Surrey. The scanning system used in this project was provided to the University of Surrey by the University College Hospital (London) and was a laser-based system used to scan human heads. A third research effort was developing a scanning system, also laser-based, at the University of Loughborough.

The specific persons and places that the CAR team would like to have visited are:

1. Sarah Cross, Defence Clothing and Textiles Authority, Colchester, Essex;
2. Stephen Cole and Jane Aspden, Stores and Clothing Research and Development Establishment, Colchester, Essex;
3. Maurice McKenna and Stephen Marshall, National Engineering Laboratory, Glasgow, Scotland;
4. Adrian Huggins, Department of Mechanical Engineering, University of Surrey, Guildford, Surrey;
5. Peter Jones, University of Loughborough, Loughborough.

At the end of May 1994 it was decided that current conditions were unfavorable for receiving permission for an on-site investigation of UK scanning devices.

Automation of Armed Forces Measurement Blank

Don O'Brien of the Defense Logistics Agency suggested that the project team might consider automating the Armed Forces Measurement Blank, which is used to handle people who cannot be fitted with one of the standard sizes. There are two versions, DD Form 358 (special sized clothing for men) and DD Form 1111 (special sized clothing for women). At the time of O'Brien's request these forms were completed manually and sent by FAX to the Defense Personnel Supply Center (DPSC). The team accepted this assignment and worked on a prototype software system. The first step was to determine the proper format of the data items to be entered on the form and find out the ranges of acceptable values. Lonnie Turner, Chief of the Clothing Issue Facility at Fort Jackson, SC, provided some information about the data items.

The procedures being conducted by DPSC at the time were observed in early October 1992, so that they could be incorporated in the software.

The requirements for the prototype system were as follows. The system should allow data to be entered interactively, with prompts for each data item and an error message if something invalid is entered. Initially the check for validity would be limited to checking format and numerical ranges of individual data items; there would not be a check of whether the person described by the measurements could be fitted with a standard size uniform (although this could probably be accomplished in a future version; if DPSC can provide an algorithm for this check it can be automated earlier). After being entered, the data are stored in a dBase IV file.

After the initial prototype was tested for feasibility, the project was to investigate a means of electrically transmitting the data to the DPSC. At the Bobbin Show 1992, the team met with Jim Della Polla and Tom Balderstone of the DPSC to discuss possibilities. Tom agreed to serve as the point of contact for this project. He indicated that alternatives for receiving data included electronic mail and telephone (via modem).

The project also investigated the possibility of providing software to allow DPSC to send the data electrically to a CAD alterations package, which DPSC hoped to install. With appropriate organization of the data, the CAD system would be able to receive computer-generated body measurements and then automatically accomplish the appropriate alterations to a standard pattern (the "blue pencil" step), and finally could direct the operation of a numerically-controlled cutter. In this way all the steps from gathering the size data to the cutting could be handled with electronic data interchange.

Also at the Bobbin Show, members of the project team met with representatives of Gerber, Inc. to discuss this part of the project. Gerber agreed to send the project team the documentation for the interface to their alterations package. With this documentation in hand, the team could determine what would be required to convert the data from the clothing form to the proper format for the alterations package.

During September 1992 a rough prototype of the interactive data entry portion of the system for DD Form 358 was developed. On October 1, 1992 Dr. Staples visited English American in Fredricksburg, MD to see how they had incorporated individuals' special measurements, provided by field representatives, directly into their AM5 CAD system for the generation of made-to-measure patterns. This was in preparation for the linkage to be made between the automated armed forces measurement blank and the CAD system being purchased by the factory at DPSC. On October 2, Dr. Staples visited the factory at DPSC to discuss the specific needs of the automated armed forces measurement blank and to observe the process then being used to handle special measurements orders. This was to aid in the development of the automated form and to prepare for the automation of the entire process.

when the new CAD system was in place. Contact persons were Colonel Bill Meadows, Jim Della Polla, and Clarence Robinson.

During October the prototype data-entry program for DD Form 358 (special-sized clothing for men) was improved. The data-entry screen then bore greater resemblance to the layout of DD Form 358. Boxes on the screen corresponded to those on the form.

To facilitate data entry, the system used the following codes:

DESCRIBE SHOULDERS:

S - Long Neck
R - Regular Neck
Q - Medium Neck
H - Short neck

DESCRIBE POSTURE:

N - Normal
E - Erect
F - Forward or Stooped
H - Half Stout
S - Stout
C - Corpulent

To help prevent data-entry errors, the data-entry program should employ range checks for selected fields. The following proposed ranges were based on data obtained from Fort Jackson.

1. Height:	58 to 80
2. Weight:	95 to 255
3. Sleeve length:	25 to 40
4. Waist:	23 to 48
5. Seat:	30 to 50
6. Breast:	30 to 50
7. Head:	19 to 25

Sarat Vemuri located what appeared to be an appropriate software package for accomplishing data transfer: Telemate. Since it was "shareware," it could be used free for 30 days on a trial basis.

The research team established contact with Mr. Clarence Robinson at DPSC, who agreed to provide guidance on this project. The next steps were to get his advice on the prototype data-entry system, and to test transfer of a file from Clemson Apparel Research to DPSC. A sample of the software was sent to DPSC and a time for the test was to be set up shortly afterwards.

The team awaited feedback from DPSC on the prototype system. After making any revisions specified by DPSC, the next step would have been to test the transfer of a file from Clemson Apparel Research to DPSC, then the system would be ready to test at Fort Jackson, SC to get feedback from the CIIP. The input from DPSC was prerequisite for further progress on this part of the project.

Members of the project team talked to Mr. Jim Della Polla of DPSC at the February 1993 Academic Apparel Research Conference. He expressed continued interest in achieving progress on this part of the project, and indicated that Sean Kelly would act as the new point of contact. Since the conference Mr. Kelly contacted members of our project. He received the materials the CAR team sent to DPSC, and was to evaluate the prototype system and provide suggestions. Nothing was ever heard in return. Later the announcement was made that the DPSC factory would be closed and it was assumed that the project was terminated.

Presentations, demonstrations, and related activities

There have been numerous occasions for the demonstration of software developed through this project. These include the following:

1993 Academic Apparel Research Conference

A paper summarizing the results of the project to date and describing the current status of this project was submitted for review and accepted for presentation at the 4th Annual Academic Apparel Research Conference held on February 8 in Raleigh, NC. The project team prepared a presentation consisting of an introduction to the problem area, an explanation of the case-based-reasoning approach, and a summary of the work to incorporate measurements from a 3D scanner. It included a demonstration of computer programs which display a three-dimensional figure and allow taking measurements including circumference, point-to-point, radial, and multiple point. As a part of this preparation, a video tape of the 3D software running on a SUN workstation was prepared. It was presented at the conference and demonstrated how the software would be used. The video also showed that measurements could be taken rather quickly when the software runs on the workstation. In addition, a PC was set up at the conference to demonstrate the software running live. The PC version does, of course, run more slowly than the SUN version, reflecting the relative power of the processors in each.

Naval and Marine Corps Reserve Officers

A demonstration of the 3D software was made on March 17, 1993 to reserve officers of the Naval and Marine Corps Reserve Center. The group was headed by Ed Hill, CDR, SC, USNR. The group was particularly interested in learning how the 3D scanning devices and

software available could help them develop a system for maintaining a database of lasts, i.e., models of human feet. Their current system maintains a large inventory of models made of wood, some of which date back to the Civil War. The objective is to develop a more efficient database on a computer so that lasts may be produced on demand but the data may be stored much more compactly on computer disk. The officers planned to include much of what they learned, including the names and addresses of two potential vendors of 3D scanning equipment, in a survey report.

AMTEX

On April 21, 1993 Dr. Staples presented a demonstration of the 3D software to Department of Energy National Labs representatives at the AMTEX meeting held at Clemson Apparel Research. The attendees will be involved in applying the technological expertise of the labs to the textile/apparel industry through the AMTEX partnership. The meeting at Clemson was a part of the process of educating the labs about the functioning of the apparel industry and about current research in the field.

German Students

On May 4, 1993 Dr. Staples presented a demonstration of the 3D software to a total of thirty faculty members and students from the University of Essen, Germany. The students were at Clemson as a part of an exchange involving Dr. Jack Kanet, Clemson Management professor and participant in CAR DLA projects.

Apparel Manufacturers

Separate demonstrations of the 3D software were presented jointly by Dr. Pargas and Dr. Staples to two major apparel manufacturers who had interest in the application of 3D non-contact body scanning to their gaining and maintaining market share. A white paper was prepared for one manufacturer, but no funds were committed. Both companies have stayed in contact for developments in the 3D project at CAR.

Videotape demonstration

A video of the current status of the 3D project was made by American Imagemakers in fall 1993. The purpose was to incorporate the 3D project in an updated version of a video describing projects being conducted at CAR. The principal investigators of the 3D project were interviewed and video clips of the software running on a SUN workstation were taken.

DLA review team

On April 19, 1994 a presentation was made at the Clemson Computer Science Department to the DLA team reviewing Clemson Apparel Research as a future demonstration site. The software demonstrated how a user could manually take measurements off of a 3D image of a person. The software also showed how the user could write and run small programs that will ultimately take measurements automatically off of a 3D image. This latter capability will allow measurements to be taken from large numbers of images, such as would be obtained from the recruit center at Fort Jackson, SC.

Tour demonstrations

Many visitors to CAR who expressed interest in the 3D software were offered the opportunity to see a demonstration.

Bobbin Show Demonstration Software

In preparation for the 1993 Bobbin Show, a PC version of the software, including the interpreter language, was created. Many problems had to be solved before the software could be ported from a SUN Workstation to a PC. These included (1) the fact that the amount of memory that a program could use under DOS was severely limited, (2) incompatibilities in the C and C++ compilers used on the SUN and PC platforms, (3) the difference in the processing speeds of the SPARC processor (running on the SUN) and the Intel processor (running on the PC), and (4) the inadequacy of the graphics library available for the PC. The effort expended in the conversion of the software should have been slight but turned out to be quite large, close to one man-month. The software was developed in time, however, and the result was a software demonstration package that was available for the 1993 Bobbin Show. The software, shown to visitors at the CAR Booth, included a demonstration of the command interpreter which allows the user to issue commands such as "selectregion, selectslice, slicemove, slicesave, selectpoint, pointmove, most, front, surfacedist, and lineardist".

Request for measurement extraction software application

On March 16, 1994 Jud Early of [TC²] met with Dr. Staples and Dr. Pargas concerning a measurement extraction module for their scanner currently under development. A list of tasks which could be begun immediately was sent to Jud on March 23. At the close of this DLA project the team had been kept up-to-date on the continuing problems in refining the [TC²] scanner (they are still having difficulty with the jump-order problem and cannot make unambiguous data sets). The team awaited word on potential collaboration with [TC²] in the use of CAR's measurement extraction software.

Accomplishments

An operational expert system

This project developed an expert system which provides a sound framework for predicting garment sizes. Since it is case-based, it really cannot fail to predict successfully if it is supplied with a sufficient number of valid cases which contain a sufficient number of measurements to predict garment sizes. Stated another way, if all measurements and garment size labels are accurate there are only two possible reasons for the system to make a wrong prediction: (1) there may be no previous case which has body measurements similar to the soldier being fitted, or (2) there may be no agreement on sizes among a number of cases which match the soldier being fitted. In the former instance there are an insufficient number of cases in the system, and in the latter case the measurements are not sufficient to predict sizes. In the latter case some additional measurement(s) could distinguish among the soldiers' bodies and help determine the proper sizes.

Demonstration of expert system feasibility

Operational tests at Fort Jackson proved the feasibility of employing an expert system in the clothing issue facility environment. Running on an ordinary IBM-compatible personal computer, the expert system predicts clothing sizes fast enough to handle soldiers without delaying clothing issue. Two computers may be required to handle the workload if measurements are manually called out and then entered in the computer as they were during operational tests, but one would probably be sufficient if a 3D scanner were employed. Measurements would be automatically sent from the scanner to the computer. If a 3D scanner-equipped system predicted with greater accuracy than human fitters, as expected, it would actually decrease overall time for clothing issue by reducing the number of try-ons.

Development of measurement extraction algorithms and command language

The original research plan included the development of interactive measurement extraction software. Due to the lack of availability of a 3D scanning device for field testing, more time was devoted to software development than originally planned and, in addition to the interactive tools, a language, ready for building automation macros, was developed.

This project accomplished much of the work necessary to integrate a 3D body scanner and size prediction expert system into the uniform-issuing process of a CIIP such as the one at Fort Jackson. The missing link remained the 3D full-body scanner. The addition of a scanner would take care of the aforementioned need for accurately taking a sufficient number of measurements for each case. The scanner would quickly capture a 3D body image. The project-produced software, which converts scanner output (a file of x-, y-, z-points) to specific body measurements would

provide the data necessary (stored cases of measurements) for the size prediction software to determine the sizes to be tried on for issue.

Stimulation of industry and government initiatives

At the start of this project there was very little activity or interest among sewn products manufacturers in 3D body scanning and its applications. Work on this project stimulated interest in, promoted discussion of, and increased visibility for the concept of 3D body scanning as applied to size prediction, made-to-measure using existing CAD technologies, and future directions in custom pattern building in 3D computer space. By the close of the project in June 1994 what had seemed a "lunatic fringe" idea when it was first suggested as a potential project in the fall 1990 had become the assumed direction for future research in apparel. The question had become "When?" not "If?"

Stimulation of academic initiatives

As an outgrowth of this project, two students at Clemson University and one at the University of North Carolina at Greensboro performed research projects for which papers were submitted in partial fulfillment of the requirements necessary to complete masters' degrees. Sarat Vemuri (Clemson) focused on the development of the expert system in "Case-Based Reasoning Applied to Uniform Size Prediction" (Appendix A). Vinit Jindal (Clemson) studied the learning curve of the case-based reasoning software in "Investigation of Learning in a Case-Based Reasoning System" (Appendix B). In cooperation with Dr. Staples, Audra Knight (UNC-G) determined the feasibility of retail applications of body scanning and size prediction in "The Market Feasibility of Body Scanning and Size Prediction Technologies at Retail" (Appendix E). In addition, three students at Clemson completed research projects as a part of the requirements of the graduate course MGT818, Management Support Systems: Dan Elias, Peter Kartawidjaja, and Raktim Sen (Appendix F). Also, four students at Clemson in the department of Computer Science were supported as graduate research assistants as a part of this project: Shan Jiang, Jasbir Menotra, Murali Earagolla, and Prahladkumar Yerra.

Technical papers submitted to national juried journals

Two technical papers were submitted by the Principal Investigators to national juried journals for review. "Automatic Measurement Extraction for Apparel from a 3D Body Scan" was submitted to the Journal of Optics and Lasers in Engineering (Appendix G). "Predicting Garment Sizes with Case-Based Reasoning" was submitted to the IEEE (Institute for Electrical and Electronics Engineers) Expert (Appendix H).

Newspaper publicity and trade publication articles

On June 3, 1993 the Greenville (SC) News published an article about the 3D project at Clemson Apparel Research (Appendix I). This was as a result of the support indicated by the Clemson University Research Foundation in their approval of funds to assist in the acquisition of a 3D body scanner (when available) for the CAR team. The August 1994 Apparel Industry Magazine featured an article on the work of Audra Knight (UNC-G graduate student) on the market feasibility of body scanning and size prediction (Appendix J). The October 1994 Apparel Industry Magazine featured an article by the Principal Investigators alerting the sewn products industry to the work they must do to prepare for the practical use of body scanning devices for size prediction and pattern development (Appendix K).

Additional Supportive Materials

Contract Data Requirements Lists (CDRLs) for the size prediction portion of this project can be found in Appendix L. Contract Data Requirements Lists (CDRLs) for the measurement extraction portion of this project can be found in Appendix M.

Appendix A

Student Thesis, Vemuri

Case Based Reasoning Applied to Uniform Size Prediction

Chandramouli Sarat Vemuri

A Scholarly Paper
Presented to the Faculty of the
Department of Computer Science
Clemson University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

December 1993
Clemson University
Clemson, S.C. 29634-1906 USA

Contents

1	Introduction	3
2	Case Based Reasoning	7
2.1	Obtaining the memory: Storing past cases	8
2.2	Indexing and Organizing	8
2.3	Retrieving and Matching	10
2.4	Adapting the retrieved case	12
2.5	Updating the memory with new case	13
3	Building knowledge base using ReMind development shell	15
3.1	The ReMind Shell	15
3.2	ReMind 'C' Library	17
3.3	Building the knowledge base	18
3.3.1	Defining a Case	18
3.3.2	Obtaining the data	18
3.3.3	Organizing Views	19
3.3.4	Clustering	19
3.3.5	Testing the cluster trees	20
3.3.6	Preparing for nearest neighbor match	20

3.4	Retrieving from the library	21
3.5	Testing the library	21
4	The Size Prediction System	23
4.1	Design of the system	23
4.2	The Prediction Engine	24
4.3	The User Interface	25
5	Conclusions	31
5.1	Limitations	31
5.2	Lessons Learned	31
5.3	Future work	32
A	Sample Initialization File	33

Chapter 1

Introduction

This paper describes the design and implementation of an Army Uniform Size prediction system. Uniforms issued at US Army Clothing Initial Issue Point are not tailored for each individual. Instead each soldier tries on one or more sizes from a set of standard sizes. These sizes to be tried on are *predicted* by fitting experts based on body measurements and body build. From the sizes tried on, a size that closely fits is selected. The close fitting size is then altered if necessary to fit the soldier.

The aim of the project is to reduce the total time taken for the prediction procedure and also to keep the cost of alteration to a minimum by selecting a closest fitting size. There are two parts to this project. One is taking measurements automatically by using a machine and the other is predicting size based on those measurements. This paper is about the later part of the project.

The automatic measurement taking system is still under development. So, the prediction system accepts manually taken measurements from user through a user interface developed under Microsoft Windows. Then the measurements are validated and are fed into the prediction engine. The prediction might take from 10 sec to a minute. Results obtained from the prediction engine are again displayed with an option to print.

The prediction engine is based on Case Based Reasoning(CBR). CBR is an approach for problem solving in which a solution for the problem at hand is adopted from the solutions of similar problems solved successfully in the past. It is very similar to an expert making decisions based on his past experience. This type of approach is particularly suitable for fields with repeating patterns of problems. The CBR approach was selected for this problem because it only needs raw data and requires no special knowledge engineering. In a field like uniform size prediction, where there are no well established "rules" or selecting criteria, and where the experts work by "intuition" rather than by using established set of rules, this approach not only is attractive but well might be the only way to go.

CBR algorithms are widely used by statisticians. Their use in knowledge engineering hasn't gained popularity because of the high computing power required to apply them to a large

set of data. Recent advancements in both CBR and computer technology have reduced this problem. Many commercial tools are available to build and manipulate knowledge base using CBR. ReMind from Cognitive Systems Inc. is one of them. ReMind contains an interactive development shell for building a knowledge base and a C library to manipulate that knowledge base.

This project focuses on *applying* available CBR techniques and tools to the problem rather than *developing* them. It will produce one of the first CBR applications intended for commercial use.

A large set of data consisting of body measurements and sizes assigned by human fitting experts corresponding to those sizes was collected from Fort Jackson SC Army base. ReMind interactive shell was used to organize and index that data and to develop a knowledge base. C programs that work on that knowledge base were written. Those C programs also present the data to the user in a consistent manner by using the Microsoft Windows graphical user interface. Given Height (in inches), Weight (in lbs), Neck, Chest, Waist, Hips and Sleeve (all in inches), the system will predict Short Sleeve Shirt Size (12.5 - 18.5), Long Sleeve Shirt Size (12.5 25 - 18.5 45), Trouser Size (25XS - 45XL), Green Coat Size (25XS - 50XL) and Black Coat Size (25XS - 50 XL). Long Sleeve Shirt Size has two components. One is the shirt size and other is the sleeve length. Trouser Size, Green Coat Size and Black Coat Size have Size and Length as their components.

The Rest of this paper is organized as follows. Chapter 2 gives an overview of Case Based Reasoning and its use in this project. Chapter 3 explains the ReMind interactive development shell and how the knowledge base is built using that shell. Chapter 4 describes the C program that manipulates the knowledge base and handles the user interface. Chapter 5 gives the conclusions and possible future work.

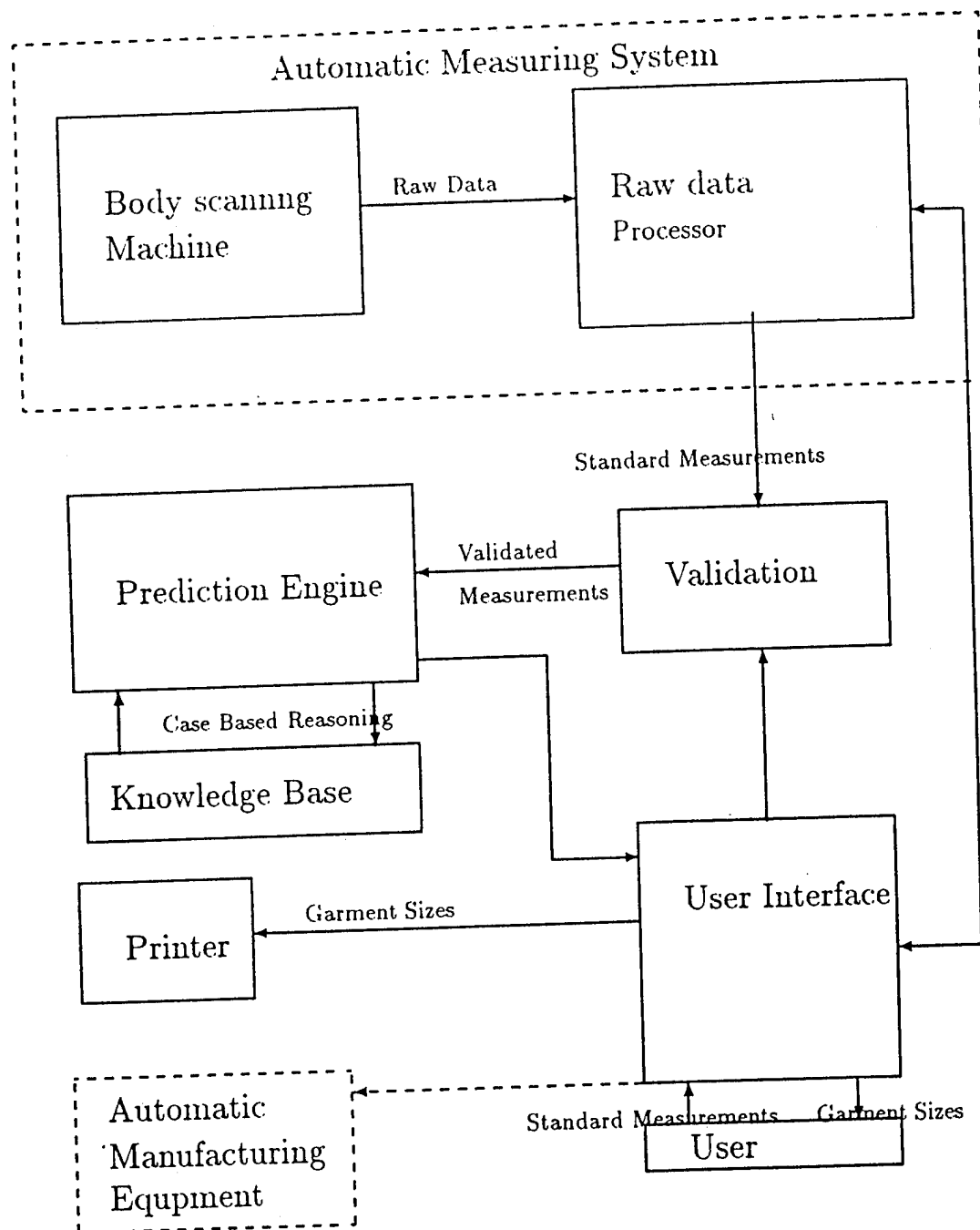


Figure 1.1: Block diagram of Size Prediction system

Chapter 2

Case Based Reasoning

It is known that humans rely on past experiences to make decisions. It seems natural to adapt the same strategy to solve problems using computers. Such strategy is called Case Based Reasoning. A case is a set of attributes that describes the problem and its solution. In CBR, the computer uses relevant stored or "remembered" cases to solve a new problem case. Instead of following an algorithmic approach to solve a problem, a case based reasoner will obtain a case of similar problem that has been solved successfully in the past and adapt the solution to the existing problem. After validating this solution and fine tuning it, this new solution is again stored and becomes one of the "remembered" cases, thus acquiring new knowledge. This cycle can continue forever in fields with widely varying problems, or it can be stopped after the knowledge base is sufficient to handle any new case without adaptation. This method of applying CBR is called a problem solving CBR. CBR can also be used to analyze the problem case by comparing it with similar previous cases as in legal or medical precedents. A solution can not be derived from previous cases, but a pro/con report can be generated for each attribute of the problem case. This method is called interpretive CBR. The size prediction project uses problem solving CBR.

Advantages of the CBR are many. The main advantage pertaining to this project is that acquiring the knowledge base or "learning" can be fairly un-complicated. Much of the knowledge needed is in the form of "cases". Thus no special knowledge engineering is required. The other advantages are, ease of generating explanations, ability to see and avoid past mistakes, capability of focusing in on the most important parts of the problem first etc.

The steps needed for a successful Case based reasoning system are: acquiring and storing a large knowledge base, organizing or indexing the knowledge base for easy retrieval, retrieving relevant cases quickly, adapting the retrieved case for the problem case and updating the knowledge base with the new case. The following sections explain the process in detail.

2.1 Obtaining the memory: Storing past cases

The performance of a CBR system depends on its knowledge base. So it is important to have a good, consistent and valid set of cases. In fields like law or medicine, where there is an existing wealth of information about past experiences, this task of acquiring knowledge base becomes easy. One can transform the library of cases into machine readable form quite easily. In other fields, one has to obtain the information gradually. CBR systems make obtaining the information very natural. One can start with a very small knowledge base. As new cases are solved using this small knowledge base, a human expert can examine the solution and correct the solution and the system adds this new solution to its knowledge base automatically. In other words, the system can "learn". This learning can be applied to fields with established case libraries as well, improving the quality of the knowledge base. Since today's mass storage devices are very inexpensive and fast, storing large cases is not a big concern. But, storing them in a fashion that facilitates easy retrieval is important. The following section describes the storage and retrieval issues.

2.2 Indexing and Organizing

The huge amount of data acquired needs to be stored in a fashion that is easy to retrieve and takes a minimum of space. That is, the data needs to be *indexed* since retrieving is essentially a search problem. This search becomes non-trivial since there is more than one key attribute in each case. Hence multi-key indexing is necessary. If the knowledge base spans a wide variety of sub-fields, it needs to be *organized*. For example, in case of legal precedents, a knowledge base might include auto liability cases and auto injury cases, which are different from each other. They need to be organized appropriately so that a retrieval in one field retrieves cases in that field only and in no other fields. In the case of size prediction, the knowledge base is homogeneous and organizing is not a problem.

There are many ways to index a knowledge base. The programmer (knowledge base builder) can fix the indices. But this will make the system unable to adapt to new domains and moreover, in some fields, the indices may not be well defined. In uniform size prediction there are no well defined indices to index the knowledge base. There are no well established rules regarding which body measurements affect each garment size. Size selection depends more or less on the "judgement" of the fitting expert and may not depend on a single set of measurements. It may be dependent on ratios of some measurements such as weight to height. So fixing indices is not suitable. Inductive learning is another approach. In inductive learning, the CBR program *clusters* the knowledge base by using algorithms. The program analyzes the data for repetitive patterns and builds relationships among outcomes and inputs. Such an approach not only is field independent but doesn't need to have defined indices. Some clustering algorithms are given by Hartigan et al[1].

The commercial development shell used in this project, ReMind, uses the clustering approach. The particular clustering algorithm used is proprietary and is not disclosed. But the algorithm is based on *Direct Splitting* and *Simultaneous Clustering and Scaling* (See [1,

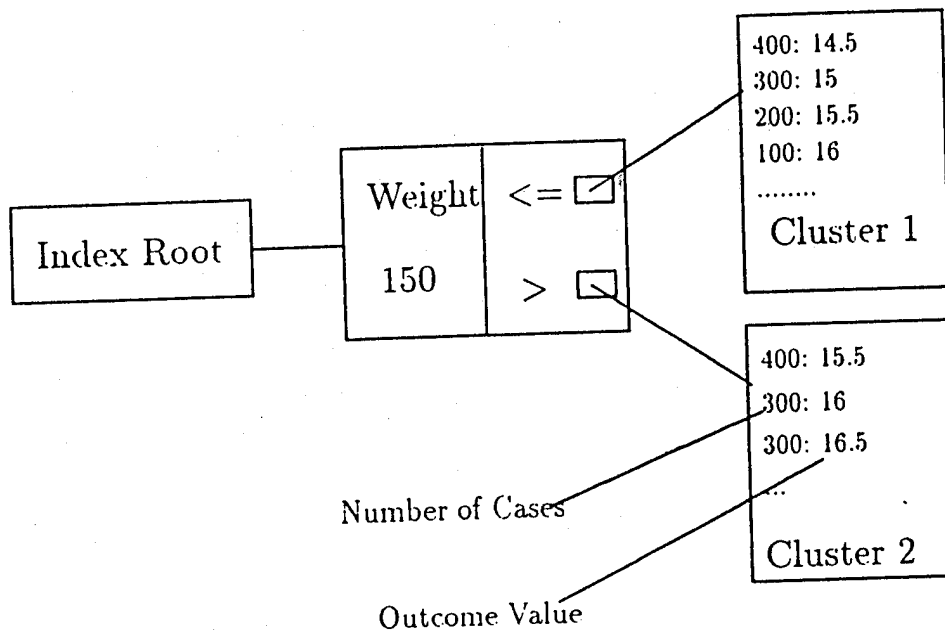


Figure 2.1: After the first split

Pages 251-278, 299-312]) and can be explained as follows.

Some fields in case are considered “match” fields and are to be designated as such by the programmer. Only those fields affect the outcome. The clustering mechanism splits the case library into groups, based on the values of the match fields provided. The algorithm considers every value of every match field in the library as a possible split. Then it will evaluate these splits and determine which split does the best job of separating the different outcomes. This the split that divides the whole library into two most homogeneous groups. This split will become a node in the cluster (binary) tree with the two groups or clusters as its children. The clustering mechanism will continue to split each cluster using the same principle recursively until the cluster is completely homogeneous or there are too few number of cases in a cluster to decide the best split. This process will result in a binary tree with tests of match fields as interior nodes and clusters as leaf nodes. Since searching a binary tree is $O(\log n)$, searching is very fast.

The following example will make the principle more clear. Consider a short sleeve shirt size which is represented as a real number ranging from 12.5 to 18.5 as the outcome field and Height, Weight, Neck, Chest and Waist as match fields. The clustering algorithm makes splits and evaluates each cluster. Suppose it decides that all the cases with Weight less than or equal to 150 will make one cluster and all others the other cluster, making the cluster look as in Figure 2.1

Then the clustering algorithm does the same thing with Cluster 1 and Cluster 2. Suppose Neck less than or equal to 15 decides the best split for Cluster 1 and Neck less than or equal to 16 for Cluster 2. Then the cluster tree looks as shown in Figure 2.2.

The clustering process continues on resulting clusters until the clusters becomes homoge-

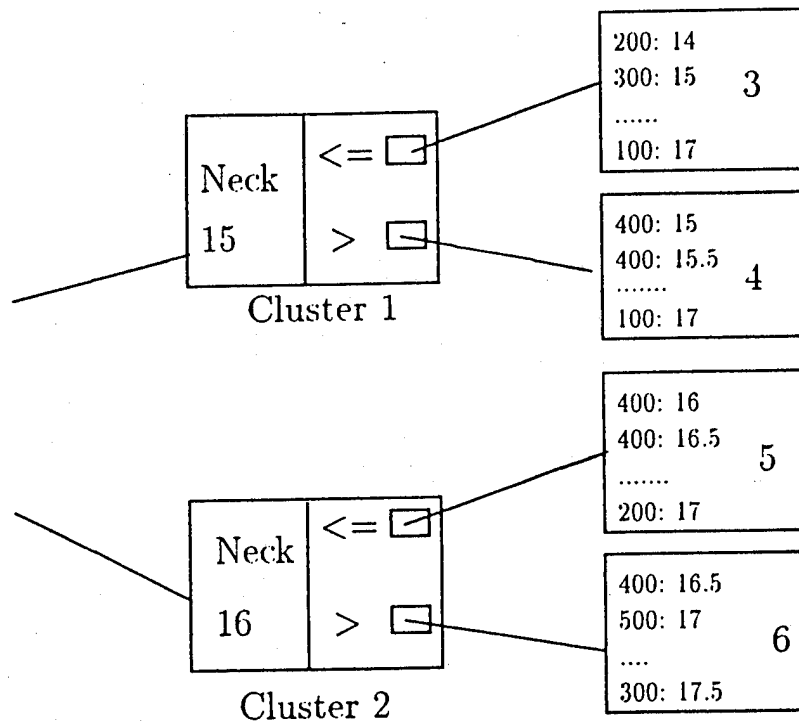


Figure 2.2: After second split

neous i.e. same outcome for all cases in that cluster or there are too few cases in the cluster. So finally the tree looks like Figure 2.3.

The clustering procedure can be done without any human interaction. But for best results in fields where the rules are well defined, a human expert can transfer some of his/her knowledge to the clustering algorithm by using a Q-Model. In a Q-Model, a person can set precedence of match fields thus changing the evaluation criteria. But for this project, this was not done because there is in-sufficient human expertise.

Above process describes the *indexing* method of ReMind. ReMind also has the capability to *organize* the case base by using prototypes. A human expert has to express domain-specific knowledge as a prototype. It is like screening the cases before clustering. For example organizing bank loans into cases related to business loans in one cluster and all cases related to car loans in other cluster.

2.3 Retrieving and Matching

The whole point of Case Based Reasoning is retrieving a case that is similar to a problem case. Many different algorithms exist for retrieving relevant cases. By using the indexing method discussed in above section, retrieving relevant cases becomes trivial. One can retrieve case(s) just by following the tree performing appropriate tests at each node to choose which branch to take. This search ultimately leads to a leaf cluster. The leaf cluster can

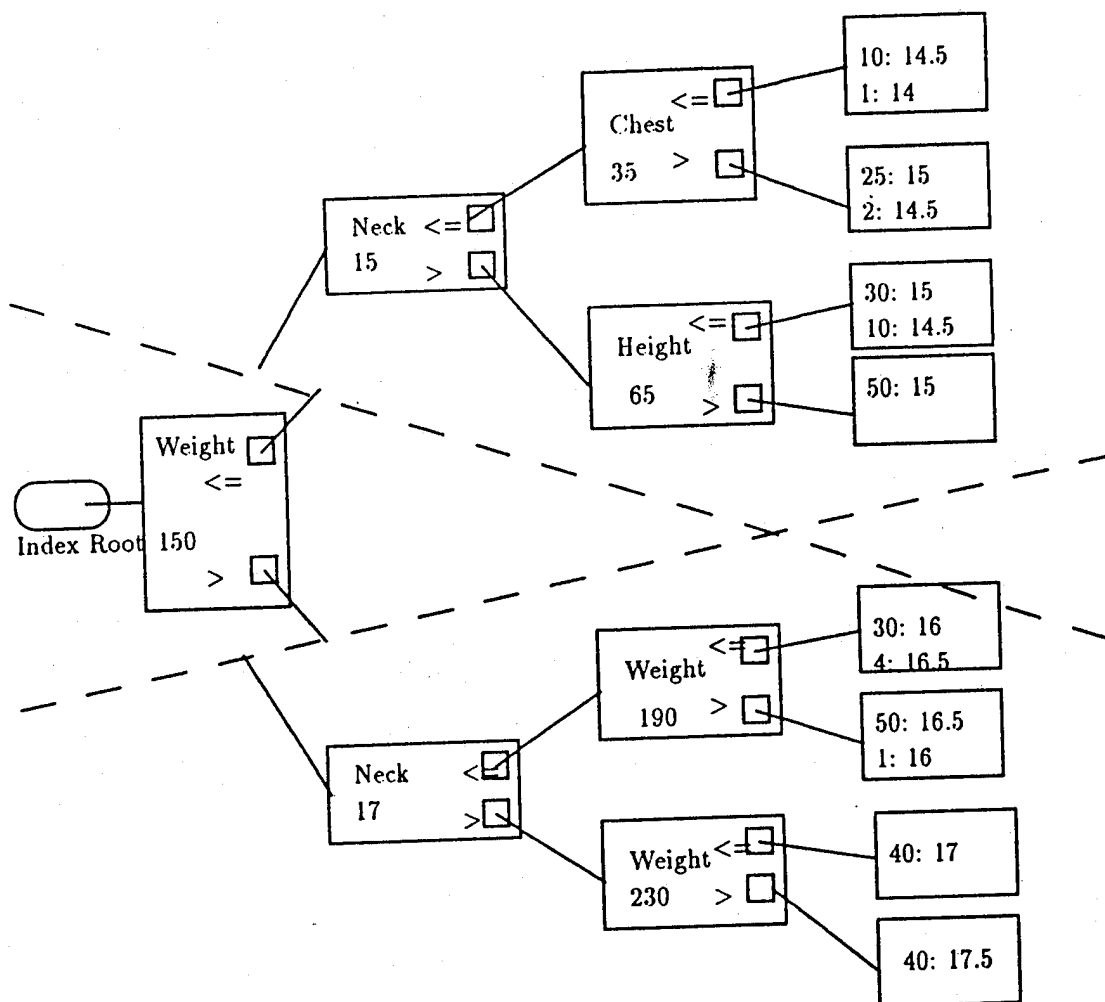


Figure 2.3: Final Tree

either consist of cases with same outcome in which case the result is clear or it may contain cases with mixed outcomes.

More often than not, one arrives at a mixed cluster. So the most similar case needs to be retrieved from this leaf cluster to determine the most similar case. This is where *matching* comes into picture. Matching is a process in which cases are compared using some measure giving each case a "similarity score". The case with highest similarity score will be the case that closely matches. There are many ways to calculate the similarity scores. And in some instances such as legal precedents, where a case is mostly textual, it might be very difficult to evaluate the case.

ReMind uses statistical method called *Nearest Neighbor Matching*. For each field in the case, a value between 0 and 100 is assigned based on the mean and standard deviation calculated for the field across the library. Then the absolute value of the difference between the input field and the examined field is calculated and subtracted from 100 to determine the similarity score for the examined case field. The programmer has to assign "weights" to each relevant field based on how important the field is in determining the outcome. The total similarity score of the case is determined by multiplying each field's similarity score and the weight of that field, and taking the average of the products. The higher the similarity value the closer the retrieved case is to the problem case.

ReMind allows this nearest neighbor method to be used on an entire library as well as on a retrieved mixed cluster. Since the algorithm is $O(n^2)$, where n is number of cases in the cluster, applying this to entire library is costly and may not be practical if the library is large. But to select the "best" case from a mixed cluster one can quickly apply the nearest neighbour algorithm.

2.4 Adapting the retrieved case

The retrieved case may not have the exact same input fields as the problem case. In some instances, it might be possible to slightly adjust the outcome value obtained from the retrieved case by using formulas or other techniques. For example in real estate pricing, if the input house has 3 bath rooms whereas the retrieved case has 2 bath rooms, it might be necessary to add some dollar amount to the total price. This technique is called adaptation, and it uses formulas. In some fields where outcomes are not quantities that can be managed by formulas, more complex methods may be needed. A human expert must determine the adaptation scheme. In the size prediction project adaptation is not applied because there is no human expert available with sufficient domain knowledge.

2.5 Updating the memory with new case

One of the strengths of Case Based Reasoning is the ability of the library to "learn". This ability is acquired by storing the new fine-tuned and solved cases in the case library. Storing a new case might require partial or complete re-indexing of the library. In some cases, updating may not be necessary when the case base is large enough to handle any input case without the need for adaptation. In other cases, where fast responses are needed, it may not be practical to update the case base with each new case. However, a batch updating can be done in those cases.

Since this size prediction project requires fast response times, it is not practical to re-index after. Also, the ReMind 'C' library does not allow for re-indexing of the library. Since the case base is large enough, there is no need for updating the library with new cases.

Chapter 3

Building knowledge base using ReMind development shell

3.1 The ReMind Shell

ReMind is a case-based reasoning development shell developed by Cognitive Systems Inc. The ReMind shell is development environment with graphical user interface based on Microsoft Windows (Macintosh and X11 versions are also available). ReMind is capable of doing all the required steps explained in the previous chapter. It has facilities to define a case base, to build a case base, to index and organize it, to interactively retrieve and to adapt the results. ReMind also has facilities to import data stored in ASCII form and to present cases in customized forms.

The ReMind development environment consists of nine editors: field editor, symbol editor, formula editor, case editor, importance editor, cluster editor, q-model editor, data import editor and form editor.

The first step in building a case base is to define all the attributes in a case which is done in *field editor*. One can interactively define a new field, define what type of the field i.e. integer, real, symbol, date etc. assign default values and other optional attributes.

Many real world problems are textual in nature. But processing text is computationally intensive. ReMind has a data type called *symbol*. A symbol similar to an enumerated type in C or Pascal. In many cases textual attributes like sizes and city names can be represented as symbols. Internally symbols are treated as numbers making symbols easy to process. The *Symbol editor* lets the programmer define the symbols used in the knowledge base. The symbols are defined in a hierarchical manner. For example a symbol *Garage-Type* might have *One-Car*, *Two-Car*, *Three-Car* and *No-Garage* as its children. So in place of *Garage-Type* one can use any of the later symbols. Also, one can enter ranking information into the symbol hierarchy using the symbol editor, making the symbols ordered. For example the symbols *Extra-Small*, *Small*, *Medium*, *Large* and *Extra-Large* can be ordered so

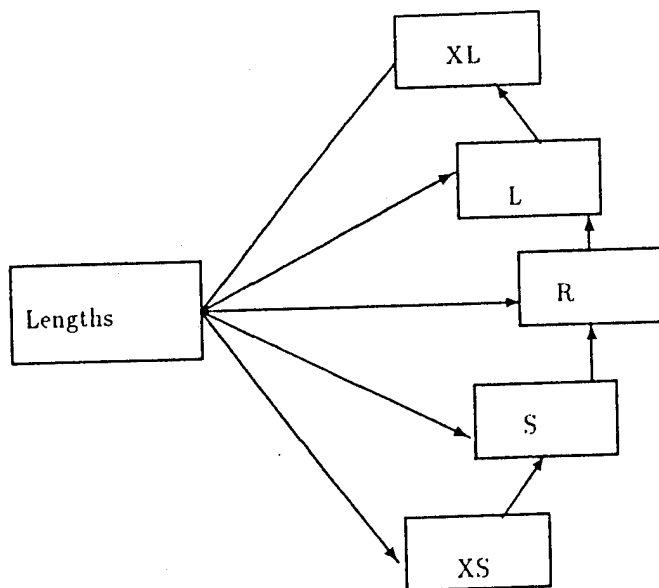


Figure 3.1: A Symbol Hierarchy

that Extra-Small is less than Small and so on. Then symbols can be used in comparisons. Figure 3.1 gives an ordered symbol hierarchy used in this project to represent garment lengths.

Some fields in the case might be calculated from other fields. The *Formula editor* can be used to define formulas using a graphical user interface. Also the formula editor checks the validity of a formula and valid type mixing.

The *Case Editor* is where the main functions of building a case library are performed. One can enter data into cases, perform retrievals, do adaptation etc. All the fields of a case are presented in a *Form filling* manner. One can input data into the case by making a New case, and typing in the form.

In indexing the case library, the system needs to know the outcome field and all the relevant "match" fields. The *Importance Editor* can be used to designate these attributes. Also in nearest neighbor matching, each relevant field needs to have weights. Assigning weights also can be done in the importance editor. Each set of weights is called a *Weight Vector*.

The *Cluster Editor* is used to index the case library. Once the cluster tree is built, the cluster editor will show the cluster tree in a graphical manner. That graphical representation can be used to selectively re-cluster parts of the cluster tree. Knowledge guided induction or clustering can be performed by using Q-Models and Prototypes.

The *Q-Model Editor* can be used to represent a human expert's knowledge graphically. This Q-Model can be used in the clustering process to do knowledge guided induction. Also, prototypes (front end screener) can be built using the Q-Model editor.

Many times, the data that is needed for building a case library is available in a machine

readable data base. Usually that data can be obtained in ASCII files. *Data Import Editor* can be used to import the data into the case library eliminating the need for re-keying the data. Conversions from one form to other eg. integer to real can be specified in this editor. Also, converting text into symbols can be done.

Form Editor can be used to create customized forms for case presentation. These customized forms are useful in highlighting important information and hiding un-necessary details.

A case library can have more than one *View*. Each view can have only one cluster tree. The Weight Vector can be view specific or it can be present in all views. Since a cluster tree is based on outcome field, and only one cluster tree can exist in one view, only one outcome field can be specified in one view. So if there is more than one independent outcome fields, different views have to be used.

There are three kinds of case "dispositions" in ReMind: stored, un-stored and hypothetical. Stored cases are ReMind's "memory". They are previously solved cases and they are used to build cluster trees and are compared in nearest neighbor matching. Un-stored cases are also solved cases. But un-stored cases are reserved for testing purposes. That is, after indexing the library, one can retrieve using an un-stored case as a problem case and compare the retrieved results with actual values in the case. ReMind provides an automatic testing function. By using this function, retrieval can be done using all un-stored cases and calculate the percentage of success. Hypothetical cases are unsolved problem cases.

Initially all solved cases in ReMind are un-stored cases. In each view, dispositions can be set randomly to reserve certain percentage as un-stored cases.

3.2 ReMind 'C' Library

The case base developed using the development shell can be accessed interactively from within the development shell. However, to achieve faster responses, to use a customized user interface to do a number of predictions automatically etc. requires the C libraries. Cognitive Systems provides a Windows Dynamic Link Library(DLL) for accessing the knowledge base built using the ReMind shell. However, there are certain limitations to this C library. One can "access" the knowledge base, but can not "modify" it. That is one can not re-index the knowledge base using the C library. This C library consists of an API (Application Program Interface) of function calls that can be used to perform almost any retrieval operation. These functions can be called from any C/C++ program to do retrievals from the case library.

Field Definitions		
Field Name	Type	Description
Measurements		
Height	Integer	Height of the person in inches
Weight	Integer	Weight of the person in lbs
Neck	Real	Neck measurement of the person in inches
Chest	Real	Chest measurement of the person in inches
Waist	Real	Waist measurement of the person in inches
Hips	Real	Hips measurement of the person in inches
Sleeve	Real	Sleeve measurement of the person in inches
Sizes		
Short Sleeve Shirt Size	Real	Size of Short Sleeve Shirt 12.5 - 18.5
Long Sleeve Shirt Size	Real	Size component of Long Sleeve Shirt 12.5 - 18.5
Long Sleeve Shirt Sleeve	Integer	Sleeve component of Long Sleeve Shirt 25 - 45
Trouser Size	Integer	Size component of Trouser 25 - 45
Trouser Length	Symbol	Length component of Trouser {XS, S, R, L, XL }
Green Coat Size	Integer	Size component of Green Coat 25 - 45
Green Coat Length	Symbol	Length component of Green Coat {XS, S, R, L, XL }
Black Coat Size	Integer	Size component of Black Coat 25 - 45
Black Coat Length	Symbol	Length component of Black Coat {XS, S, R, L, XL }

Table 3.1: Field Definitions

3.3 Building the knowledge base

3.3.1 Defining a Case

The first step in building a knowledge base is to define a case using the field editor. Table 3.1 describes different fields and their types.

Notice that Long Sleeve Shirt, Trouser, Green Coat and Black Coat sizes are divided into two parts each. That is because, the two parts are not exactly dependent on the same factors and have to be predicted separately. For example, Trouser Size is mostly dependant on body build, waist and hips where as Trouser Length is dependant on height. Also size is numeric and length is a symbol. So they are separated into two fields.

3.3.2 Obtaining the data

Any case library needs a good set of solved cases as its "memory". In this project, these solved cases are obtained as follows. In Fort Jackson, SC Army base, approximately 600 soldiers per week are fitted by human experts. A army standard form is used to record measurements and garment sizes of each soldier. With the co-operation of the Clothing Initial Issue Point at Fort Jackson, the project team was able to obtain copies of these forms. Approximately 4000 forms were obtained. With the help of people at Clemson

Apparel Research, those 3000 forms were converted into dBase (A PC based database) records. Each record consists of all the data needed for each case i.e. measurements and garment sizes. After obtaining all the 3000 records, a dBase program was written by the project team to weed out bad cases resulting from typographical errors. dBase records can be exported to standard ASCII files. Since only ASCII data can be imported into ReMind, all 3000 records were exported into an ASCII file.

Symbols XS,S,R,L,XL were defined using the symbol editor and they were ordered so that $XS < S < R < L < XL$. In Data Import editor, conversion formulas were defined to convert from ASCII data to integers, reals. Lengths like XS, S, R, L, XL were converted into symbols previously defined.

3.3.3 Organizing Views

As stated earlier, each view can contain only one cluster tree thus requiring one view per component of garment to predicted. So after importing the data into the case library, nine views were defined corresponding to the nine garment components namely: Short sleeve shirt size, Long sleeve shirt size, Long sleeve shirt sleeve, Trouser size, Trouser length, Green coat size, Green coat length, Black coat size and Black coat length. In each view, dispositions of cases are set as 95% stored and 5% un-stored. These un-stored cases will be used in testing the library.

3.3.4 Clustering

The next step in building a case library is to index the data. As explained in previous chapter, ReMind uses *Clustering* to index data. There are nine garment components to be predicted. So there are nine different cluster trees to be built in nine different views. An *outcome* field and all the *match* fields affecting the outcome field needs to be defined for each cluster tree. The importance editor should be used to designate the fields in each view. Table 3.2 lists all the views and outcome and match fields in each view.

After the fields are designated appropriately, cluster trees need to be built. Building cluster tree is done in the cluster editor. No knowledge guided induction i.e. using Q-Models was done. One parameter, *minimum cases to split*, needs to be given to the clustering algorithm. The clustering algorithm will not try to split a cluster if there are fewer than this minimum number of cases in that cluster preventing over-clustering. Over-clustering can lead to meaningless evaluation of cluster by the algorithm, since there are too few cases to represent all aspects. This parameter was set at 15 after trying a number of alternatives. Once this parameter is set, the rest of the clustering process is automatic.

Field designations in each view		
View name	Outcome field	Match fields
Short Sleeve Shirt Size	Short Sleeve Shirt Size	Weight, Height, Neck, Chest, Waist, Sleeve
Long Sleeve Shirt Size	Long Sleeve Shirt Size	Weight, Height, Neck, Chest, Waist, Sleeve
Long Sleeve Shirt Sleeve	Long Sleeve Shirt Sleeve	Weight, Height, Neck, Chest, Waist, Sleeve
Trouser Size	Trouser Size	Weight, Height, Waist, Hips
Trouser Length	Trouser Length	Weight, Height, Waist, Hips
Green Coat Size	Green Coat Size	Weight, Height, Neck, Chest, Waist, Hips, Sleeve
Green Coat Length	Green Coat Length	Weight, Height, Neck, Chest, Waist, Hips, Sleeve
Black Coat Size	Black Coat Size	Weight, Height, Neck, Chest, Waist, Hips, Sleeve
Black Coat Length	Black Coat Length	Weight, Height, Neck, Chest, Waist, Hips, Sleeve

Table 3.2: Outcome and Match fields in each view

3.3.5 Testing the cluster trees

The *un-stored* cases can be used to test the performance of the cluster tree. ReMind provides a *Test Clusters* command to the testing automatically. When this command is invoked, ReMind will consider each un-stored case as a problem case and try to solve it by retrieving. Since the outcome is already known, the retrieved outcome is compared to actual outcome and results are reported. If more than 80% of un-stored cases have an outcome similar to the retrieved outcome, the cluster tree is considered good.

3.3.6 Preparing for nearest neighbor match

Although doing nearest neighbor matches on large libraries is time consuming, applying them to do the small subset of cases retrieved by an inductive retrieval yields good results. For doing nearest neighbor match, a *Weight Vector* needs to be defined. A weight vector is a set of weighings attached to each relevant field. Since each garment component has different relevant fields, different weight vectors need to be defined. This is done in the importance editor. Each field can be given a weight of 0,2,4,8 or 16. A weight of 4 is twice as important as a weight of 2 and so on. Table 3.3 describes weight vectors in the case library

Weight Vectors for each outcome field							
Weight Vector	Weights						
	Height	Weight	Neck	Chest	Waist	Hips	Sleeve
Short Sleeve Shirt Size	1	16	16	8	1	0	16
Long Sleeve Shirt Size	1	16	16	8	1	0	0
Long Sleeve Shirt Sleeve	8	16	4	1	1	0	0
Trouser Size	0	8	0	0	16	16	0
Trouser Length	16	8	0	0	8	8	0
Green Coat Size	1	16	16	8	1	0	16
Green Coat Length	1	8	2	2	2	2	0
Black Coat Size	1	16	16	8	1	0	16
Black Coat Length	1	8	2	2	2	2	0

Table 3.3: Weight Vectors

3.4 Retrieving from the library

Once the library is built and indexed, it can be used for retrieval. But there are many ways it can be used. One can do a simple inductive retrieval or simple nearest neighbor match. Simple inductive retrieval can yield a mixed cluster where more than one outcome is retrieved. Simple nearest neighbor match can be time consuming. So the following approach is adapted.

- A minimum of twenty cases are retrieved using inductive retrieval.
- A nearest neighbor match using appropriate weight vector is done among those retrieved case to obtain ten cases that are similar to the problem case.
- Voting is done among those ten outcomes to arrive at first, second and third choices

Interactive retrievals from within the development shell can be done from the case editor. for predicting a garment component size, body measurements are entered into a new case in case editor. There are menu choices to invoke the inductive retrieval and then nearest neighbor match on the retrieved cases. But there is no way to do the voting. So when using the development shell to retrieve, voting is not done and cases with highest similarity scores are taken as first, second and third choices. But when using the C library provided, voting can be done. So the voting method was used in developing the size prediction system using the C library and programs written to test the library. The size prediction system is explained in later chapters and the testing program is described in the following section.

3.5 Testing the library

A C program was written to predict all the un-stored cases in the library for a given garment component using the retrieval strategy discussed in the above section. Figure 3.2 gives the

-
1. For each un-stored case in the library do
 2. Retrieve 20 cases using inductive retrieval
 3. Do a nearest neighbor match on those 20 cases to obtain 10 most similar cases
 4. Vote amount the 10 cases to obtain first, second and third choices. Here voting is just head-count.
 5. Print actual outcome and the first, second and third choice outcomes and which one matched the actual outcome. Increment the counter corresponding to that choice to indicate the number of matches in that choice
 6. EndDo
 7. Print total number of cases predicted. Number of matches for each choice and percentages

Figure 3.2: Testing Algorithm

algorithm of the testing program.

This program takes an un-stored case, determines the first, second and third choice outcomes and compares them with the actual outcome. It repeats the process with all the un-stored cases in the library and produces the total of how many first choices were correct, how many second choices were correct and how many third choices were correct and calculates the percentages.

The results indicate that the first choice was correct in about 80% of the cases, 10-15% second choice, 5-10% third choices and 1-5% the actual outcome was not in the top three choices predicted. See the conclusions for discussion on results.

Chapter 4

The Size Prediction System

This chapter describes the C/C++ program that manipulates the knowledge base built using the ReMind development shell. This program uses the ReMind C libraries.

Requirements for the size prediction system are as follows. The system should take input from the automatic measuring equipment or from the user by using an GUI. It needs to predict all the garment sizes, doing multiple retrievals if necessary. Then the predictions should be presented to the user using a GUI and the system should be able to print the predictions on custom form. The ReMind interactive development shell can not be used to build the size prediction system because of the above requirements. The user interface of the development shell is not flexible. Multiple retrievals can not be done without user interaction. Printing capabilities are limited in the development shell. So there is no alternative other than to use the C libraries. But since ReMind will not allow modification of the knowledge base using the C libraries, the "learning" capabilities of a CBR system are not exploited. This may not be an issue since the case library is large and close to being complete.

4.1 Design of the system

Figure 4.1 describes the design of the system. There are two independent components to the system: the user interface and the prediction engine. The software is implemented using the Borland C++ version 3.1 and requires Microsoft Windows 3.x. Windows 3.x is also required by the ReMind development shell and C libraries.

The user interface is responsible for accepting the input from the user, validating it and supplying the data to the prediction engine. It also presents the results to the user and is able to print the results. The prediction engine is responsible for dealing with the knowledge base built using the ReMind development shell, doing retrievals as necessary by following the strategy explained in previous chapter and accomplishing the logging if required.

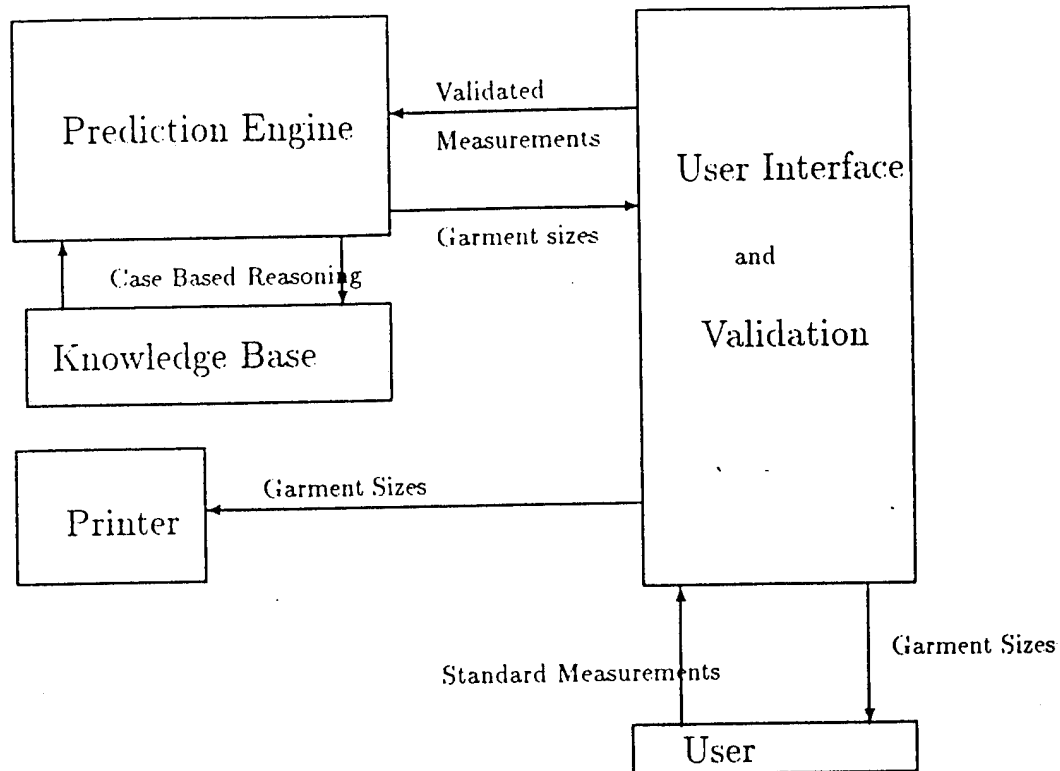


Figure 4.1: Design of the system

The user interface and the prediction engine are independent of each other. The design allows for the user interface to be changed without affecting the prediction engine. The user interface can be replaced with a program that collects data from the automatic measuring equipment once that capability is available.

4.2 The Prediction Engine

The prediction engine is implemented using the ReMind C library and is written in C. It resides in module `predict.c`. This module is responsible for interacting with the case library. The prediction engine consists of three visible functions. They are `InitSizepCBR`, `ShutdownSizepCBR` and `PredictCBR`.

`InitSizepCBR` will take init file name as an argument and initializes the ReMind system, opens the library, determines the measurement field ids and associates view name, outcome field, weight vector relationship. Figure 4.2 contains the algorithm for this function. Appendix A gives a sample initialization file.

`ShutdownSizepCBR` will stop the remind system, close the library and log files. It also deletes the blank case created by the initialization function.

-
1. Open the initialization file. If can not be opened stop.
 2. Read library file name and log file name from the init file.
 3. Read measurement field names from init file. Eg. Weight, Height etc.
 4. Read view name, outcome field, weight vector relations from init file.
 5. Initialize ReMind system and open library and log files
 6. Determine the internal field-ids of the measurement fields from the library for the given names.
 7. Establish view handle, outcome field-id, weight vector handle relations using the relations read from the init file and by using the library.
 8. Create a new blank case in the library. This case is filled with the measurements to be predicted.

Figure 4.2: Algorithm for InitSizepCBR

PredictSizepCBR is the core of the prediction engine. This function takes a structure filled with measurements, and two function pointers. The structure has three empty strings. These strings will be filled with first, second and third choice outcomes. Each string contains the garment components short sleeve shirt size, long sleeve shirt size, long sleeve shirt sleeve, trouser size, trouser length, green coat size, green coat length, black coat size and black coat length in that order. The function pointers are called at the beginning and end of each component retrieval. The user interface displays the progress of prediction using these function pointers. This function uses the retrieval strategy described in the previous chapter. Figure 4.3 gives the algorithm for this function.

Notice that the prediction module is not dependant on the way the input measurements were obtained. Once the automatic measurement system is available, the inputs can be obtained directly from the machine instead of using the user interface.

4.3 The User Interface

The user interface is based on the Microsoft Windows Graphical User Interface and uses the object oriented class libraries in BC++ 3.1. It is developed using the Borland C++ 3.1 and ObjectWindows class library. ObjectWindows hides the details of the Windows API by using a set of classes. The user interface elements like dialog boxes, push buttons etc. are constructed using the Resource Workshop that comes with BC++ 3.1

Figure 4.4 gives the object model used by the user interface. TDialog and TApplication are

-
1. Fill new case allocated by the initialization function with the measurements using the structure passed in.
 2. For each view, outcome field, weight vector relation DO
 3. Retrieve 20 cases using inductive retrieval.
 4. Among those 20 cases do a nearest neighbor match to get 10 most similar cases.
 5. Vote among those 10 cases to determine first, second and third choice outcomes.
 6. Log the results and append the the outcomes to the first second and third choice strings in the transfer structure passed in.
 7. EndDO.

Figure 4.3: PredictSizeCBR Algorithm

classes in the ObjectWindows library. TDialog provides a convenient interface for manipulating the dialog boxes. TApplication is basically a place holder and every application that uses ObjectWindows needs to have a class derived from that class. TSizepApplication is derived from TApplication. Its constructor invokes the function InitSizepCBR to initialize the prediction engine.

There are two classes TSizepWindow and TPredictDialog derived from TDialog. TSizepWindow is program's main window. The purpose of this window is to accept measurements from the user and allow the user to start prediction. Figure 4.5 shows the main window as displayed on the screen. The main windows consists of edit boxes for name of the person and all the measurements. It also contains push buttons "Predict" for starting prediction and "Clear" for clearing all the edit boxes. TsizepWindow contains functions for validating the input measurements. When the push button "Predict" is pressed, it is validated and the prediction engine is invoked by filling the measurements in a structure and by passing to PredictSizepCBR. Once the results are returned by the prediction engine, an instance of the class TPredictDialog is constructed by passing all the measurements and the results to that class.

TPredictDialog presents the results in a window as shown in Figure 4.6. This window contains static text boxes for name, all the measurements and predicted garment sizes. It also has "Print" and "OK" buttons. Pushing "Print" button causes the results to be printed on a printer that is connected to "LPT1". Printing in MS Windows is very lengthy because of its WYSWYG requirement. Since only simple text output is needed for this project, direct printing to the printer port is done. Pushing "OK" causes the window to destroy and the control returns to TSizepWindow.

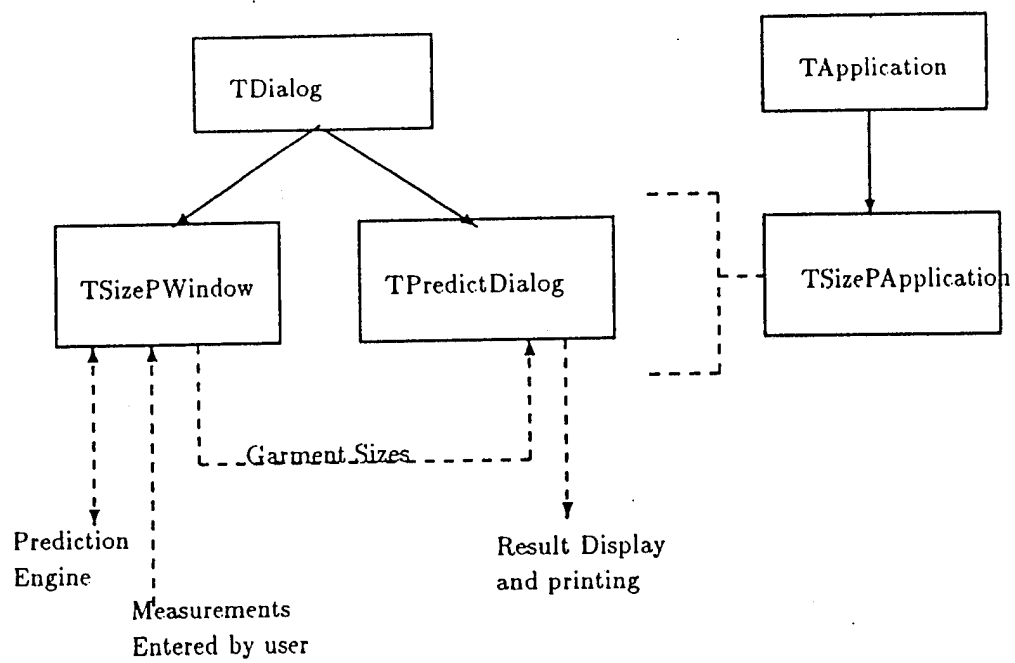


Figure 4.4: Object Model

Size Prediction

 **Clemson Apparel Research
Size Prediction System**

First Name Last Name

Height in

Weight lbs

Head in

Neck in

Sleeve in

Waist in

Hips in

Chest in

Figure 4.5: The main Window

Predictions					
Clemson Apparel Research Size Prediction Results					
Last Name		VEMURI		First Name	SARAT
		1st Choice	2nd Choice	3rd Choice	
Short Sleeve Shirt		15.5	16	15	
Long Sleeve Shirt		15 33	15.5 34	14.5 35	
Black Coat		40 L_	38 L_	38 L_	
Green Coat		38 L_	39 L_	40 L_	
Trouser		31 L_	32 R_	33 R_	
Cap					
Height	72 in	Waist		32 in	
Head	17 in	Hips		34 in	
Neck	15 in	Sleeve		34 in	
Chest	35 in	Weight		165 lbs	
Ok		Print			

Figure 4.6: Results Window

Chapter 5

Conclusions

5.1 Limitations

For 75 provided accurate results. However, a case library is only as good as the data from which it was built. Since fitting is a very subjective field and since a fitters choice of a particular size varies depending on human perceptions, fatigue and other immeasurable variables, the case library is not as consistent as we would like it to be. Also, the garment sizes are sometimes inaccurate. For example a size 15.5 short sleeve shirt might be slightly larger than it should be and hence it might be chosen instead of a size 16.0. The measurements also depend on who is taking them. The same person can be measured differently by different measurers or even by the same measurer. All these variables are beyond our control but will affect the case library. Once the automatic measuring equipment is available, the inconsistency in the measurements should be eliminated. Also, as improvements in garment manufacturing are made, garment sizes will become more consistent. The only remaining variable is the expert fitters subjectiveness. If the system is changed to "learn" as time goes on, these fitters can be aided by the size prediction system thus making the garment size choices consistent.

5.2 Lessons Learned

This project was started about one and half years ago and there are many lessons learned as we worked on this project. Some of them are:

- A large set of cases are needed for a case library to work optimally. At least 50 cases per outcome are recommended for accurate prediction.
- A "good" set of cases is need for consistent predictions. The limitations explained in the previous section result from a lack of consistency in the data obtained.

- Case based reasoning can be very slow. Because of the large amounts of data involved, considerable computing power and processor memory is required. The size prediction system is usable only on high end IBM compatible personal computers. However, as technological advancements in computing industry are occurring at a rapid pace, computing power becomes less of a limitation.

5.3 Future work

A first step toward improvement would be minimizing affects of the limitations discussed above by fine-tuning the system. Also, a number of enhancements to the size prediction system are possible. Here are some of them.

1. A customized clustering algorithm may improve the clustering process and speed of retrievals.
2. Online help can be added to the user interface.
3. The user interface can be enhanced so that it can be used by a computer illiterate person. The intended user base is mostly computer illiterate. So this can be a big step in making the system more adaptable.
4. The clustering process builds a binary cluster tree with comparisons as nodes and outcomes as leaf nodes. This is like a big if-then-else structure which is typical of a rule-based expert system. One can investigate the possibility of using case based reasoning to build a rule-based expert system. Some work has already been done in this direction. But because ReMind does not have a convenient method of accessing the cluster tree from a C program, developement was temporarily halted.

Appendix A

Sample Initialization File

```
#####  
##  
## This is the initialization file for the size prediction  
## project.  
## This must exist for proper operation of the program  
##  
#####  
  
###  
### NOTE:  
### If an identifier contains whitespace in its name  
### enclose it in double quotes  
###  
  
# The following contains the CBR library to operate upon  
# CBRLIBNAME is the KEY. The format is  
# CBRLIBNAME libraryfullpath  
CBRLIBNAME \CBRLIBS\JACK.CBR  
  
# The following contains the LOG file name  
# CBRLOGNAME is the KEY. The format is  
# CBRLOGNAME logfullpath  
# Set logfullpath to NILL if no log needed  
CBRLOGNAME \CBRPROGS\logfile  
  
# The following measurement fields MUST be specified. KEY words are  
# WEIGHT, HEIGHT, NECK, SLEEVE, CHEST, HIPS, WAIST  
# in any order. Format is  
# KEYWORD fieldname
```

WEIGHT Weight
HEIGHT Height
NECK Neck
SLEEVE Sleeve
CHEST Chest
HIPS Hips
WAIST Waist

The following triplets contains fieldname, view, weightvector
predicted.
All the following outcome fields should be specified in the
same order.
You can precede a triplet by the keyword NOPREDICT to
make the system to ignore the entry. But the entry should
be there.

Short Sleeve Shirt Size
#NOPREDICT
"Short Sleeve Shirt Size" "Short Sleeve Shirt Size" "Short Sleeve Shirt Size"
Long Sleeve Shirt Size
#NOPREDICT
"Long Sleeve Shirt Size" "Long Sleeve Shirt Size" "Long Sleeve Shirt Size"
Long Sleeve Shirt Sleeve
#NOPREDICT
"Long Sleeve Shirt Sleeve" "Long Sleeve Shirt Sleeve" "Long Sleeve Shirt Sleeve"
Trouser Size
#NOPREDICT
"Trouser Size" "Trouser Size" "Trouser Size"
Trouser Length
#NOPREDICT
"Trouser Length" "Trouser Length" "Trouser Length"
Green Coat Size
#NOPREDICT
"Green Coat Size" "Green Coat Size" "Green Coat Size"
Green Coat Length
#NOPREDICT
"Green Coat Length" "Green Coat Length" "Green Coat Length"
Black Coat Size
#NOPREDICT
"Black Coat Size" "Black Coat Size" "Black Coat Size"
#Black Coat Length
#NOPREDICT
"Black Coat Length" "Black Coat Length" "Black Coat Length"

Bibliography

- [1] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.
- [2] Cognitive Systems, Inc. *ReMind Developer's Reference Manual*. Cognitive Systems, Inc., 1992.
- [3] Cognitive Systems, Inc. *ReMind API draft*, Cognitive systems Inc. 1992.
- [4] Simoudis Evangelos. "Using Case-Based Retrieval for Customer Technical Support", *IEEE Expert*, October 1992, IEEE Press.
- [5] Hennessy D, Hinkle D, "Applying Case-Based Reasoning to Autoclave Loading", *IEEE Expert*, October 1992, IEEE Press.
- [6] Pearce et al, "Case Based Design Support", *IEEE Expert*, October 1992, IEEE Press.
- [7] Case Based Reasoning, *Proceedings of a Workshop on Case Based Reasoning, Pensacola Beach FL, May 1989*, Morgan Kaufmann Publishers, Inc.
- [8] Kolodner J and Riesbeck C, *Experience, Memory and Reasoning*, Lawrence Erlbaum Associates, Publishers, 1986.
- [9] Technical reports for DLA project DLA900-87-D-0017, DO 0026.
- [10] Borland Inc., Borland C++ 3.1 programmers reference manual.
- [11] Borland Inc., ObjectWindows 3.0 reference manual.
- [12] Norton P and Yao P, *Borland C++ Programming for Windows*, Bantom Books, 1993.
- [13] Norton P and Yao P, *Windows 3.0 Programming Techniques*, Bantom Books, 1992.

Appendix B

Student Thesis, Jindal

Investigation of learning in a Case-Based Reasoning System

Vinit Jindal
Department of Computer Science
Clemson University

Abstract

An expert system is just a computer program that uses knowledge and inference procedures to solve problems just as a human expert would do. Inference procedures employed in expert systems can broadly be divided into two classes: Rule based and Case based. Case-based systems simplify knowledge acquisition. It is not necessary to determine how experts reason, instead, one just needs to gather a number of solved cases. Little research has been published on determining how performance increases as a function of the number of cases and on determining how many cases are 'enough'. Our objective was to evaluate the influence of learning on performance of a specific case-based reasoning system. We adopted a case-based reasoning system for predicting garment sizes based on the body measurements. We started with a small knowledge base and performed test runs on the system while progressively adding cases into the knowledge base. The test runs included performing garment size predictions for a set of test cases and comparing the predictions with the actual size issued for that body measurement. The results of the test runs were analyzed and 'learning curves' for different garments were plotted. We found that learning occurred in the system at a very early stage and subsequent addition of knowledge made little change in the system performance.

Investigation of learning in a Case-Based Reasoning System

Vinit Jindal

A Scholarly Paper
Presented to the Faculty of the
Department of Computer Science
Clemson University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

August 1994
Clemson University
Clemson, SC 29631 USA

Acknowledgements

I feel a sincere urge to recall and acknowledge with deep gratitude, the inspiration provided to me by **Dr. Steve Davis** for taking up this project. His valuable guidance throughout the project in overcoming the hurdles of designing and analysis is beyond the reach of my expression. I would also like to express a deep gratitude towards **Dr. Jack Peck** and **Dr. Mike Westall** for their time and valuable suggestions on the proceedings of this project.

This acknowledgement would remain incomplete without the mention of **Ms. Renee' Lambert**. I would like to thank her for encouraging me throughout the life of the project and providing me with all the necessary facilities of the Information Systems Lab. I would also like to thank **Anurag Shrivastava** and **Sudhakar Rao** for being patient with me and freeing me to work on this project. Finally, I express my thanks to **Ms. Lea Benson** and **Ms. Kelly Bearden** at Computer Science for providing me with all the information needed for project completion and presentation.

Contents

1. Introduction	1
2. Case Based Reasoning	4
2.1 Obtaining the Memory	5
2.2 Indexing and Organizing	5
2.3 Retrieving and Matching	7
2.4 Adapting the Retrieved Case	8
2.5 Updating the Memory	9
3. Building the Knowledge Base	11
3.1 The ReMind Shell	11
3.2 ReMind 'C' Library	13
3.3 Building the Knoilwdge Base	14
3.3.1 Defining a Case	14
3.3.2 Obtaining the Data	15
3.3.3 Organizing Views	15
3.3.4 Clustering	16
3.3.5 Preparing for Nearest Neighbor Match	16
3.3.6 Prepating for Template Retrieval	17
3.4 Verifying the Library	17
4. Testing the Knowledge Base	19
4.1 Design of the Testing System	19
4.1.1 The Search Strategy	20
4.1.2 Test Cases	21
4.1.3 Adding Knowledge to the Knowledge Base	21
4.1.4 User Interface	22
4.2 Results	22

5 Conclusion	33
5.1 Limitations	33
5.2 Lessons Learned	34
 Appendix A	
The Test Program Source Code	36
 Appendix B	
Test Results with 10 Nearest Neighbor Search	45
 Bibliography	49

List of Figures

2.1 Clustering: After the first split	6
2.2 Clustering: After the second split	7
3.1 Symbol hierarchy.....	12
3.2 Case field definitions	14
3.3 Weight vector definitions	16
4.1 Test procedure algorithm	20
4.2 Prediction summary for short sleeve shirt size	24
4.3 Learning curves for short sleeve shirt size	25
4.4 Learning curves for short sleeve shirt size (stacked chart)	26
4.5 Learning curves for short sleeve shirt size (true scale)	27
4.6 Prediction summary for trouser size.....	28
4.7 Learning curves for trouser size	29
4.8 Learning curves for trouser size (stacked chart)	30
4.9 Learning curves for trouser size (true scale)	31
B-1 Summary for short sleeve shirt size (10 nearest neighbors)	45
B-2 Curves for short sleeve shirt size (10 nearest neighbors)	46
B-3 Curves for short sleeve shirt size (10 nearest neighbors stacked chart)	47

Chapter 1

Introduction

This paper is an attempt to evaluate the influence of learning on performance of a Case-based expert system. An expert system is just a computer program that uses knowledge and inference procedures to solve problems just as a human expert would do. The inference procedure can broadly be divided into two classes: Rule-based procedure and Case-based procedure. Although Rule-based systems are the most popular type, Case-based systems have some advantages and are beginning to be used for commercial applications. Case-based systems simplify knowledge acquisition.

Case-Based Reasoning (CBR) is an approach for problem solving in which a solution for a problem at hand is adopted from the solutions of similar problems solved successfully in the past. It is very similar to an expert making a decision based on his past experience. This type of approach is particularly suitable for situations with repeating patterns of problems. The CBR approach only needs raw data about previously solved problems and requires no special knowledge engineering. In areas where there are no well established rules and where human experts work by "intuition" rather than established rules, this approach not only is attractive but well might be the only way to go.

CBR algorithms are widely used by statisticians. Their use in knowledge engineering hasn't gained popularity because of the high computing power required to apply them to a large set of data. Recent advancements in both CBR and computer technology have reduced this problem. Several commercial tools are now available to build and manipulate knowledge base using CBR. ReMind from Cognitive Systems Inc. is one of them. ReMind contains an interactive development shell for building a knowledge base and a C library to manipulate that knowledge base.

The first question that comes to mind for a case based system is: How many cases would be enough for desired performance ? Having more than necessary cases is undesirable due to:

1. Time taken to collect the cases.
2. Time and resources involved to enter them into the system.
3. Increase in search time.

one would expect a Case-based reasoning system to "learn" (or improve performance) as the cases are added, as has been shown in previous studies by Bareiss[2] and Aamodt [3]. Study of the literature indicates that little research has been published on determining how performance increases as a function of the number of cases and on determining how many cases are "enough".

The aim of this study is to evaluate the influence of learning on performance of a specific case based reasoning system, and if possible to develop general guidelines for such evaluation. The focus here is to evaluate performance of an available CBR application rather than developing one.

As a testbed, we adopted a Case-based reasoning system for predicting garment size developed for an ongoing project at Clemson Apparel Research Center. The system has about 4000 cases of body measurement of soldiers together with the garment size issued. These cases are used to predict the garment size for a given body measurement.

The rest of this paper is organized as follows: Chapter 2 gives an overview of Case-based reasoning, Chapter 3 explains the ReMind interactive development shell and how the knowledge base is built using that shell, Chapter 4 describes the testing strategy, the test runs and results, Chapter 5 gives the conclusion and future work.

Chapter 2

Case Based Reasoning

It is known that humans rely on past experiences to make decisions. It seems natural to adapt the same strategy to solve problems using computers. Such strategy is called Case Based Reasoning (CBR). A case is a set of attributes that describes the problem and its solution. In CBR, the computer uses relevant stored or 'remembered' cases to solve a new problem case. Instead of following an algorithmic approach to solve a problem, a case based reasoner will obtain a case of similar problem that has been solved successfully in the past and adapt the solution to the existing problem. After validating this solution and fine tuning it, this new solution is again stored and becomes one of the 'remembered' cases, thus acquiring new knowledge. This cycle can continue forever in fields with widely varying problems, or it can be stopped after the knowledge base is sufficient to handle any new case without adaptation. This method of applying CBR is called a problem solving CBR. CBR can also be used to analyze the problem case by comparing it with similar previous cases as in legal or medical precedents. A solution can not necessarily be derived from previous cases, but a pro/con report can be generated for each attribute of the problem case. This method is called interpretive CBR. The size prediction project uses problem solving CBR.

Advantages of the CBR are many. The main advantage pertaining to this project is that acquiring the knowledge base or 'learning' can be fairly uncomplicated. Much of the knowledge needed is in the form of 'cases'. Thus no special knowledge engineering is required. The other advantages are: ease of generating explanations, ability to see and avoid past mistakes, capability of focusing in on the most important parts of the problem first etc.

The steps needed for a successful Case based reasoning system are: acquiring and storing a large knowledge base, organizing or indexing the knowledge base for easy retrieval, retrieving relevant cases quickly, adapting the retrieved case for the problem case and updating the knowledge base with the new case. The following sections explain the process in detail.

2.1 Obtaining the memory: Storing past cases

The performance of a CBR system depends on its knowledge base. So it is important to have a good, consistent and valid set of cases. In fields like law or medicine, where there is an existing wealth of information about past experiences, this task of acquiring a knowledge base becomes easy. One can transform the library of cases into machine readable form quite easily. In other fields, one has to obtain the information gradually. CBR systems make obtaining the information very natural. One can start with a very small knowledge base. As new cases are solved using this small knowledge base, a human expert can examine the solution and correct the solution and the system adds this new solution to its knowledge base automatically. In other words, the system can "learn". This learning can be applied to fields with established case libraries as well, improving the quality of the knowledge base. Since today's mass storage devices are very inexpensive and fast, storing large cases is not a big concern. But, storing them in a fashion that facilitates easy retrieval is important. The following section describes the storage and retrieval issues.

2.2 Indexing and Organizing

The huge amount of data acquired needs to be stored in a fashion that is easy to retrieve and takes a minimum of space. That is, the data needs to be indexed since retrieving is essentially a search problem. This search becomes non-trivial since there is more than one key attribute in each case. Hence multi-key indexing is necessary. If the knowledge base spans a wide variety of sub-fields, it needs to be organized. For example, in case of legal precedents, a knowledge base might include auto liability cases and auto injury cases, which are different from each other. They need to be organized appropriately so that a retrieval in one field retrieves cases in that field only and in no other fields. In the case of size prediction, the knowledge base is homogeneous and organizing is not a problem.

There are many ways to index a knowledge base. The programmer (knowledge base builder) can fix the indices. But this will make the system unable to adapt to new domains and moreover, in some fields, the indices may not be well defined. In uniform size prediction there are no well defined indices to index the knowledge base. There are no well established rules regarding which body measurements affect each garment size. Size selection depends more or less on the "judgement" of the fitting expert and may not depend on a single set of measurements. It may be dependent on ratios of some measurements such as weight to height. So fixing indices is not suitable. Inductive learning is another approach. In inductive learning, the CBR program clusters the knowledge base by using algorithms. The program analyzes the data for repetitive patterns and builds relationships among outcomes and inputs. Such an approach is independent of the application area and doesn't need to have defined indices. Some clustering algorithms are given by Hartigan et al[1].

The commercial development shell used in this project, ReMind, uses the clustering approach. The particular clustering algorithm used is proprietary and is not disclosed. But

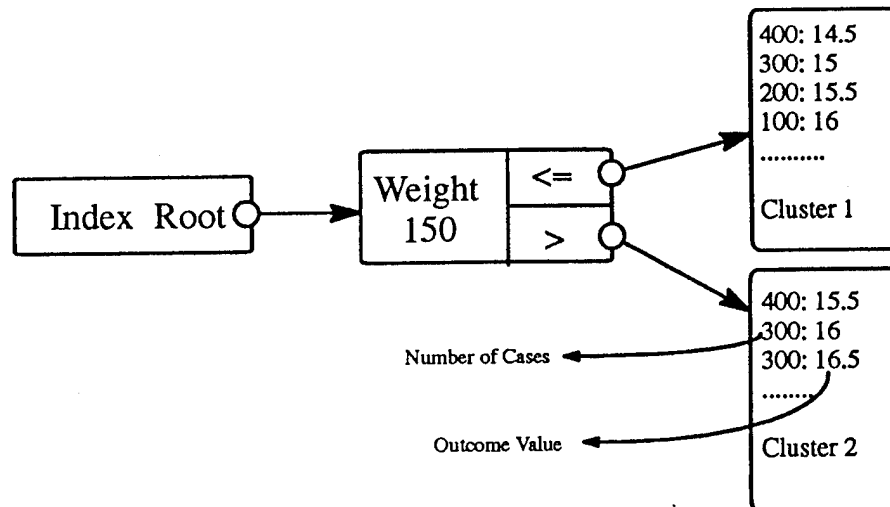


Figure 2.1 After the first split

the algorithm is based on Direct Splitting and Simultaneous Clustering and Scaling (See [1, Pages 251–278, 299–312]) and can be explained as follows.

Some fields in case are considered "match" fields and are to be designated as such by the programmer. Only those fields affect the outcome. The clustering mechanism splits the case library into groups, based on the values of the match fields provided. The algorithm considers every value of every match field in the library as a possible split. Then it will evaluate these splits and determine which split does the best job of separating the different outcomes. This the split that divides the whole library into two most homogeneous groups. This split will become a node in the cluster (binary) tree with the two groups or clusters as its children. The clustering mechanism will continue to split each cluster using the same principle recursively until the cluster is completely homogeneous or there are too few number of cases in a cluster to decide the best split. This process will result in a binary tree with tests of match fields as interior nodes and dusters as leaf nodes. Since searching a binary tree is $O(\log n)$, searching is very fast.

The following example will make the principle more clear. Consider a short sleeve shirt size which is represented as a real number ranging from 12.5 to 18.5 as the outcome field and Height, Weight, Neck, Chest and Waist as match fields. The clustering algorithm makes splits and evaluates each cluster. Suppose it decides that all the cases with Weight less than or equal to 150 will make one cluster and all others the other cluster, making the cluster look as in Figure 2.1. The clusters have been selected such that the sizes are generally larger in one cluster than in the other.

Then the clustering algorithm does the same thing with Cluster 1 and Cluster 2. Suppose Neck less than or equal to 15 decides the best split for Cluster 1 and Neck less than or equal to 16 for Cluster 2. Then the cluster tree looks as shown in Figure 2.2. The clustering process

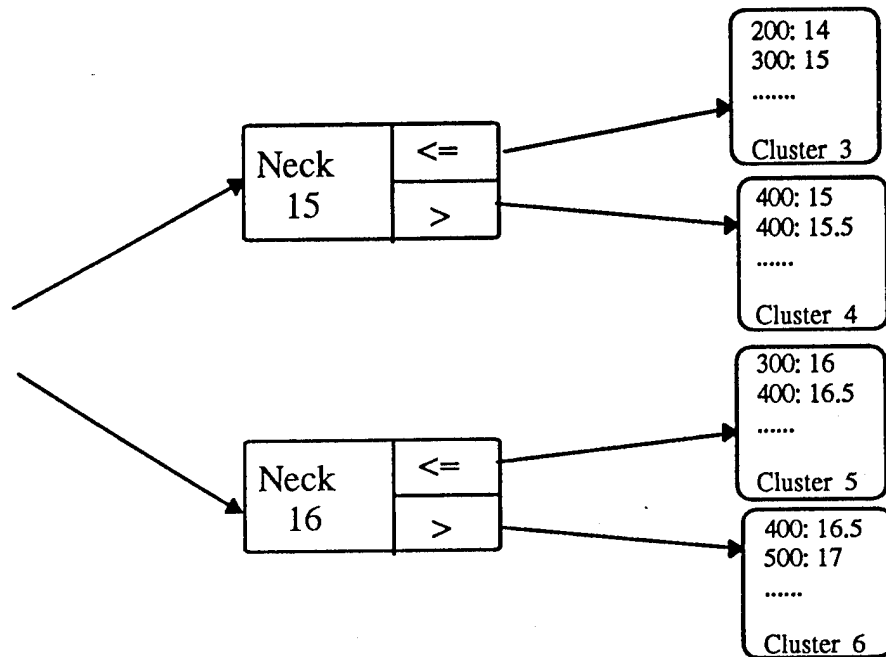


Figure 2.2: After second split

continues on resulting clusters until the clusters becomes homogeneous i.e. same outcome for all cases in that cluster or there are too few cases in the cluster

The clustering procedure can be done without any human interaction. But for best results in fields where the rules are well defined, a human expert can transfer some of his/her knowledge to the clustering algorithm by using a Q-Model. In a Q-Model, a person can set precedence of match fields thus changing the evaluation criteria.

The above process describes the indexing method of ReMind. ReMind also has the capability to organize the case base by using prototypes. A human expert has to express domain-specific knowledge as a prototype. It is like screening the cases before clustering. For example organizing bank loans into cases related to business loans in one cluster and all cases related to car loans in other cluster.

2.3 Retrieving and Matching

The whole point of Case Based Reasoning is retrieving a case that is similar to a problem case. Many different algorithms exist for retrieving relevant cases. By using the indexing method discussed in above section, retrieving relevant cases becomes trivial. One can retrieve case(s) just by following the tree performing appropriate tests at each node to

choose which branch to take. This search ultimately leads to a leaf cluster. The leaf cluster can either consists of cases with same outcome in which case the result is clear or it may contain cases with mixed outcomes.

More often than not, one arrives at a mixed cluster. So the most similar case needs to be retrieved from this leaf cluster to determine the most similar case. This is where matching comes into picture. Matching is a process in which cases are compared using some measure giving each case 'similarity score'. The case with highest similarity score will be the case that closely matches. There are many ways to calculate the similarity scores. And in some instances such as legal precedents, where a case is mostly textual, it might be very difficult to evaluate the case.

ReMind uses a statistical method called Nearest Neighbor Matching. For each field in the case, a value between 0 and 100 is assigned based on the mean and standard deviation calculated for the field across the library. Then the absolute value of the difference between the input field and the examined field is calculated and subtracted from 100 to determine the similarity score for the examined case field. The programmer has to assign 'weights' to each relevant field based on how important the field is in determining the outcome. The total similarity score of the case is determined by multiplying each field's similarity score and the weight of that field, and taking the average of the products. The higher the similarity value the closer the retrieved case is to the problem case.

ReMind allows this nearest neighbor method to be used on an entire library as well as on a retrieved mixed cluster. Since the algorithm is $O(n^2)$, where n is number of cases in the cluster, applying this to entire library is costly and may not be practical if the library is large. But to select the 'best' case from a mixed cluster one can quickly apply the nearest neighbor algorithm.

2.4 Adapting the retrieved case

The retrieved case may not have the exact same input fields as the problem case. In some instances, it might be possible to slightly adjust the outcome value obtained retrieved case by using formulas or other techniques. For example in real estate if the input house has 3 bath rooms and the retrieved case has only 2 bath rooms, it is necessary to add some dollar amount to the total price. This technique is called adaptation, and it uses formulas. In some fields where outcomes are not quantities that can be managed by formulas, more complex methods may be needed. A human expert must determine the adaptation scheme. In the size prediction project adaptation is not applied because there is no human expert available with sufficient domain knowledge.

2.5 Updating the memory with new cases

One of the strengths of Case Based Reasoning is the ability to 'learn'. This ability is acquired by storing the new fine-tuned and solved cases in the library. Storing a new case might require partial or complete re-indexing of the library. In some cases, updating may not be necessary when the case base is large enough to handle any input case without the need for adaptation. In other cases, where fast response is needed, it may not be practical to update the case base with each new case. However, a batch updating may prove to be more practical in such cases.

Chapter 3

Building the knowledge base

3.1 The ReMind Shell

ReMind is a case-based reasoning development shell developed by Cognitive Systems Inc. The ReMind shell is a development environment with graphical user interface based on Microsoft Windows (Macintosh and X11 versions are also available). ReMind is capable of doing all the required steps explained in the previous chapter. It has facilities to define a case base, to build a case base, to index and organize it, to interactively retrieve and to adapt the results. ReMind also has facilities to import data stored in ASCII form and to present cases in customized forms.

The ReMind development environment consists of nine editors: field editor, symbol editor, formula editor, case editor, importance editor, cluster editor, q-model editor, data import editor and form editor.

The first step in building a case base is to define all the attributes in a case which is done in the field editor. One can interactively define a new field, define what type of the field i.e. integer, real, symbol, date etc. assign default values and other optional attributes.

Many real world problems are textual in nature. But processing text is computationally intensive. ReMind has a data type called symbol. A symbol similar to an enumerated type in C or Pascal. In many cases textual attributes like sizes and city names can be represented as symbols. Internally symbols are treated as numbers making symbols easy to process. The Symbol editor lets the programmer define the symbols used in the knowledge base. The symbols are defined in a hierarchical manner. For example a symbol Garage-Type might have One-Car, Two-Car, Three-Car and No-Garage as its children. So in place of Garage-Type one can use any of the later symbols. Also, one can enter ranking information into the symbol hierarchy using the symbol editor, making the symbols ordered. For example the symbols Extra-Small, Small, Medium, Large and Extra-Large can be ordered so that

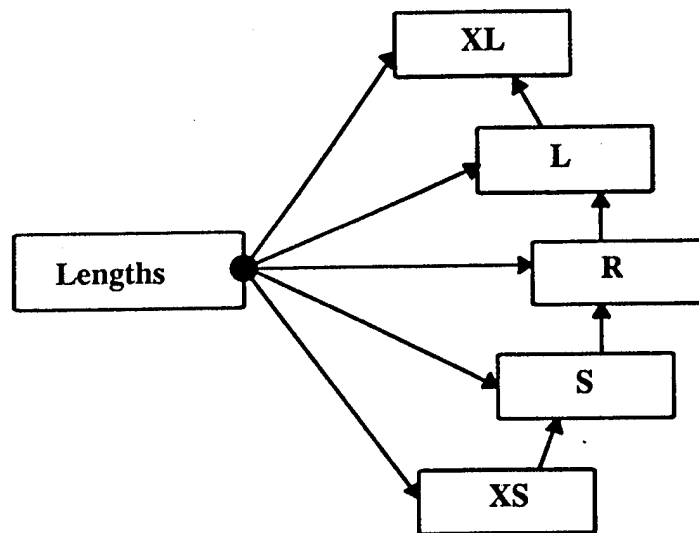


Figure 3.1: Symbol hierarchy

Extra-Small is less than Small and so on. Then symbols can be used in comparisons. Figure 3.1 gives an ordered symbol hierarchy used in this project to represent garment lengths.

Some fields in the case might be calculated from other fields. The Formula editor can be used to define formulas using a graphical user interface. Also the formula editor checks the validity of a formula and valid type mixing.

The Case Editor is where the main functions of building a case library are performed. One can enter data into cases, perform retrievals, do adaptation etc. All the fields of a case are presented in a *form filling* manner. One can input data into the case by making a 'New case', and typing in the form.

In indexing the case library, the system needs to know the outcome field and all the relevant "match" fields. The Importance Editor can be used to designate these attributes. Also in nearest neighbor matching, each relevant field needs to have weights. Assigning weights also can be done in the importance editor. Each set of weights is called a *Weight Vector*.

The Cluster Editor is used to index the case library. Once the cluster tree is built, the cluster editor will show the cluster tree in a graphical manner. That graphical representation can be used to selectively re-cluster parts of the cluster tree. Knowledge guided induction or clustering can be performed by using Q-Models and Prototypes.

The Q-Model Editor can be used to represent a human expert's knowledge graphically. This Q-Model can be used in the clustering process to do knowledge guided induction. Also, prototypes (front end screener) can be built using the Q-Model editor.

Many times, the data that is needed for building a case library is available in a machine readable data base. Usually that data can be obtained in ASCII files. The Data Import Editor can be used to import the data into the case library eliminating the need for re-keying the data. Conversions from one form to other eg. integer to real can be specified in this editor. Also, converting text into symbols can be done.

The Form editor can be used to create customized forms for case presentation. These customized forms are useful in highlighting important information and hiding unnecessary details.

A case library can have more than one view. Each view can have only one cluster tree. The Weight Vector can be view specific or it can be present in all views. Since a cluster tree is based on outcome field, and only one cluster tree can exist in one view, only one outcome field can be specified in one view. So if there is more than one independent outcome fields, different views have to be used.

There are three kinds of case 'dispositions' in ReMind: stored, un-stored and hypothetical. Stored cases are ReMind's "memory". They are previously solved cases and they are used to build cluster trees and are compared in nearest neighbor matching. Un-stored cases are also solved cases. But un-stored cases are reserved for testing purposes. That is, after indexing the library, one can retrieve using an un-stored case as a problem case and compare the retrieved results with actual values in the case. ReMind provides an automatic testing function. By using this function, retrieval can be done using all un-stored cases and calculate the percentage of success. Hypothetical cases are unsolved problem cases.

Initially all solved cases in ReMind are un-stored cases. In each view, dispositions can be set randomly to reserve a certain percentage as un-stored cases.

3.2 ReMind 'C' Library

The case base developed using the development shell can be accessed interactively from within the development shell. However, to achieve faster responses, to use a customized user interface to do a number of predictions automatically etc. requires the C libraries. Cognitive Systems provides a Windows Dynamic Link Library(DLL) for accessing the knowledge base using the ReMind shell. However, there are certain limitations to this C library. One can 'access' the knowledge base, but can not 'modify' it. That is, one can not re-index the knowledge base using the C library. This C library consists of an API (Application Program Interface) of function calls that can be used to perform almost any retrieval operation. These functions can be called from any C/C++ program to do retrievals from the case library.

Field Name	Type	Description
Case Number	Integer	Unique ID for each case.
Measurements:		
Height	Integer	Height of the person in inches
Weight	Integer	Weight of the person in lbs
Neck	Real	Neck measurement of the person in inches
Chest	Real	Chest measurement of the person in inches
Waist	Real	Waist measurement of the person in inches
Hips	Real	Hips measurement of the person in inches
Sleeve	Real	Sleeve measurement of the person in inches
Sizes:		
Short Sleeve Shirt Size	Real	Size of Short Sleeve Shirt 12.5 – 18.5
Long Sleeve Shirt Size	Real	Size component of Long Sleeve Shirt 12.5 – 18.5
Long Sleeve Shirt Sleeve	Integer	Sleeve component of Long Sleeve Shirt 25 – 45
Trouser Size	Integer	Size component of Trouser 25 – 45
Trouser Length	Symbol	Length component of Trouser { XS, S, R, L, XL }
Green Coat Size	Integer	Size component of Green Coat 2.5 45
Green Coat Length	Symbol	Length component of Green Coat {XS, S, R,L, XL }
Black Coat Size	Integer	Size component of Black Coat 25 – 45
Black Coat Length	Symbol	Length component of Black Coat {XS, S, R, L,XL }

Table 3.1: Field Definitions

3.3 Building the knowledge base

Initially, we had decided to use the knowledge base built for a previous research at Clemson Apparel Research Center. But for additional control over each record, we had to improvise upon the knowledge base. A new knowledge base was built based on the existing one.

3.3.1 Defining a Case

The first step in building a knowledge base is to define a case using the field editor. Table 3.1 describes different fields and their types.

Notice that Long Sleeve Shirt, Trouser, Green Coat and Black Coat sizes are divided into two parts each. That is because, the two parts are not exactly dependent on the same factors and have to be predicted separately. For example, Trouser Size is mostly dependant on body

build, waist and hips where as Trouser Length is dependant on height. Also size is numeric and length is a symbol. So they are separated into two fields.

3.3.2 Obtaining the data

Any case library needs a good set of solved cases as its "memory". In this project, these solved cases are obtained as follows. At the Fort Jackson, SC Army base, approximately 600 soldiers per week are fitted by human experts. An army standard form is used to record measurements and garment sizes of each soldier. With the co-operation of the Clothing Initial Issue Point at Fort Jackson, the team working on another ongoing research at Clemson Apparel Research Center, was able to obtain copies of these forms. Approximately 4000 forms were obtained. With the help of people at Clemson Apparel Research, those 4000 forms were converted into dBase (A PC based database) records. Each record consists of all the data needed for each case i.e. measurements and garment sizes. After obtaining all the 4000 records, a dBase program was written by the project team to weed out bad cases resulting from typographical errors. These cases were then shuffled randomly to eliminate any particular kind of ordering in them. DBase records can be exported to standard ASCII files. Since only ASCII data can be imported into ReMind, exactly 4056 records were exported into an ASCII file.

Symbols XS,S,R,L,XL were defined using the symbol editor and they were ordered so that $XS < S < R < L < XL$. In Data Import editor, conversion formulas were defined to convert from ASCII data to integers, reals. Lengths like XS, S, R, L, XL were converted into symbols previously defined.

3.3.3 Organizing Views

As stated earlier, each view can contain only one outcome field thus requiring one view per component of garment to predicted. So after importing the data into the case library, nine views were defined corresponding to the nine garment components namely: Short sleeve shirt size, Long sleeve shirt size, Long sleeve shirt sleeve, Trouser size, Trouser length, Green coat size, Green coat length, Black coat size and Black coat length. In each view, dispositions of first 100 cases (the test cases) were left to hypothetical and of the rest were set to un-stored (discussed in detail in next chapter). These hypothetical cases were tested against rest of the library.

Weight Vectors for each outcome field							
Weight Vector	Weights						
	Height	Weight	Neck	Chest	Waist	Hips	Sleeve
Short Sleeve Shirt Size	1	16	16	8	1	0	16
Long Sleeve Shirt Size	1	16	16	8	1	0	0
Long Sleeve Shirt Sleeve	8	16	4	1	1	0	0
Trouser Size	0	8	0	0	16	16	0
Trouser Length	16	8	0	0	8	8	0
Green Coat Size	1	16	16	8	1	0	16
Green Coat Length	1	8	2	2	2	2	0
Black Coat Size	1	16	16	8	1	0	16
Black Coat Length	1	8	2	2	2	2	0

Table 3.2: Weight Vectors

3.3.4 Clustering

The next step in building a case library would be to index the data. As explained in the previous chapter, ReMind uses Clustering to index data. There are nine garment components to be predicted. So nine different cluster trees should be built in nine different views. An outcome field and all the match fields affecting the outcome field needs should be defined for each cluster tree. The importance editor should be used to designate the fields in each view. Building cluster tree can be done in the cluster editor. This project used Nearest Neighbor search instead of Inductive search (for reasons explained in next chapter). Nearest Neighbor search uses only Weight Vectors (and no cluster trees). So no cluster tree was built for this knowledge base.

3.3.5 Preparing for nearest neighbor match

For doing nearest neighbor match, a Weight Vector needs to be defined. A weight vector is a set of weighing attached to each relevant field. Since each garment component has different relevant fields, different weight vectors need to be defined. This is done in the importance editor. Each field can be given a weight of 0,2,4,8 or 16. A weight of 4 is twice as important as a weight of 2 and so on. Table 3.2 describes weight vectors in the case library.

3.3.5 Preparing for template retrieval

In addition to Inductive and Nearest Neighbor retrieval, cases can also be retrieved by Template retrieval. Template retrieval is akin to a simple conditional search based on a Template. A Template is a list of specific criteria for retrieving cases in ReMind. For instance a template might say: 'Trouser Size ≥ 25 AND Black coat length = S'. A search on this template comes up with all the cases satisfying both the condition – Trouser Size greater than or equal to 25 and Black coat length equal to Small.

Template retrieval was extensively used to change the number of stored cases in the library. Several templates were declared on-the-fly depending on the need of the tests performed on the library. A template can be defined by using 'New Template' option in the Case editor menu. The templates used in this project were defined to retrieve a different set of cases based on their case number. All of them had a form: 'Case Number $> x$ AND Case Number $\leq y$ ' where x and y were some integers between 1 and 4056. The templates were named ' x to y ' where x and y were integers used in that template. Template retrieval was used only to make different sets of cases, not for prediction.

3.4 Verifying the library

Once the library is built, it can be used for retrieval. But there are many ways it can be used. One can do a simple inductive retrieval or simple nearest neighbor match. Simple inductive retrieval can yield a mixed cluster where more than one outcome is retrieved. Since this library doesn't have any cluster trees built, inductive retrieval was not possible.

A preliminary test on the library was performed by simple nearest neighbor search through the ReMind interactive shell. A case was randomly selected from the hypothetical cases and a nearest neighbor search was performed using one of the defined weight vectors. The results were checked intuitively and were cross-checked with the results obtained from the knowledge base from a previous research (mentioned in section 3.3). Several such retrievals were performed using different weight vectors. All results were found satisfactory. The library is now ready for the final test.

chapter 4

Testing the knowledge base

This chapter describes the procedure used to evaluate CBR system learning. It includes results for the knowledge base built using the ReMind interactive shell. Also, it describes the test program which uses the ReMind C libraries.

To evaluate the influence of learning on performance of the system, the basic function of the test procedure were as follows: Select an outcome field, for instance 'Short sleeve shirt size'. Mark a certain number of cases as test cases and keep them separate from the rest of the library. These cases cannot be marked as stored and should not participate in prediction process. Mark some cases as 'stored' from the rest of the library and perform prediction for each of the test cases against these stored cases (let's call this one run of prediction). Increase the number of stored cases in the library and perform another run of prediction. Repeat the prediction runs with increasing number of stored cases until a run is performed with all the cases in the library marked 'stored'. Record all the results and repeat the entire procedure for another outcome field. Figure 4.1 gives the algorithm for the test procedure.

4.1 Design of the testing system

From the very beginning, the testing system was designed to completely automate the test procedure thus reducing chances of human error. This however required use of some tool other than the ReMind Interactive Shell. Cognitive Systems provides a Windows Dynamic Link Library (DLL) for accessing the knowledge base built using the interactive shell. The DLL really consists of C/C++ libraries with an array of API (Application Program Interface) function calls to perform various operations on the knowledge base.

-
- Initialize the Library, make a list of hypothetical cases – the test cases.
 - Select first template in the library.
 - While template exists DO
 - Make list of cases using this template – the list of active cases in the library.
 - Open a new file for the results for this template.
 - For each of the test cases DO
 - Perform a nearest neighbor search from the active case list.
 - Vote on these nearest neighbors and pick out the top three outcomes.
 - Write the results in the opened file.
 - End DO.
 - Close the results file.
 - Select next template.
 - End DO.
 - Close the Library.

Figure 4.1: Test Procedure Algorithm

Although these API functions provide fairly complete control over the knowledge base, there are functions that can be performed only through the interactive shell and not through the API. For instance one could add knowledge to the knowledge base or change the status of cases to 'stored' or 'un-stored' but cannot re-index (or re-cluster) the knowledge base using the API library.

4.1.1 The Search strategy

The limitations of the API posed some problem in the testing system design. Inductive retrieval requires clustering (or indexing) to perform search. For new knowledge to come into effect, re-indexing must be performed before an inductive retrieval. This means for inductive search, every time new knowledge is added into the system the knowledge base must be re-indexed – a process possible only through the interactive shell.

An alternative to inductive search is nearest neighbor search. Although nearest neighbor search may yield results better than inductive search, since it exhaustively searches the entire case library, applying it to large libraries is very time consuming and impractical for commercial applications. However, we selected this time consuming procedure because with it we could set up a testing system which could operate in a batch mode, conducting a number of test runs without human intervention.

4.1.2 Test cases

The system requires setting aside a set of test cases that would not be a part of the case library. The first 100 cases in the library were chosen to be the test cases. To avoid any kind of biasing, records in the data file were randomly sorted before importing them in the knowledge base. Then we chose the first 100 consecutive cases as the test cases. Thus the test cases were randomly selected.

4.1.3 'Adding knowledge' to the knowledge base

To test learning, the system must 'add knowledge' to the knowledge base after each prediction run. But instead of physically adding knowledge to the knowledge base each time, we used a simpler and more efficient procedure. All the cases were added to the knowledge base when it was initially built using the interactive shell. At run-time, instead of using all the cases available in the knowledge base for prediction, the system selectively makes a list of cases from the available cases and performs the prediction run only on that list. For successive runs, the number of cases in this list is increased the procedure is repeated.

This procedure required some mechanism to select a list of cases to be active in the library just by specifying how many cases we want. This was accomplished by using template retrieval over the field 'Case Number'. Case Number is a unique integer (starting from 1 and increments of 1) serving as the ID for a case. To make a list of 100 cases for instance, we use the template 'Case Number > 100 AND Case Number <= 200' (because first 100 cases are the test cases!). Similarly for making a case list of 500 cases, we use template 'Case Number > 100 AND Case Number <= 600'.

The API posed no significant limitation on declaring templates dynamically, but we chose to pre-define all the templates through the interactive shell for ease of programming. It is only a one time job anyway.

4.1.4 User Interface

Since there was no human interaction required, the test procedure has no user interface module. All input is given through command line parameters and all results are written in data files. However, the system does display on screen a progress report to show what point in execution it is at currently. This was important since a typical set of runs for one outcome field lasted anywhere between 20 to 33 hours.

During a set of runs, the system opened (and closed) a new file for each run. This was done for a couple of reasons: first, to keep the results of each run separate, and second, to make available partial results of a set even if the system crashed amidst a set.

The command line input parameters for the system included: library name (knowledge base to work on), view name (case disposition and outcome field are dependent on view), and weight vector name (to perform nearest neighbor search). For our tests, we used two variants of the test program: one which finds 5 nearest neighbors for each case, and second which finds 10 nearest neighbors.

4.2 Results

A single set of results for an outcome field typically contained results from 19 prediction runs – each for a different number of stored cases in the knowledge base; and a single prediction run produced the result of prediction for each of the 100 test cases. The prediction for each of the cases included the case number, actual size, and predicted size (of the garment in consideration). Instead of making a single prediction, the system votes among 5 (or 10) nearest neighbors and depending on the vote, comes up with three choices – first, second and third predictions. The results from each of these test runs were summarized to show how many of prediction 1, prediction 2 and prediction 3 matched the actual size and how many predictions were outright misses.

Figure 4.2 shows the summarized results for outcome field ‘Short Sleeve Shirt Size’. The first column is the test run number; the second column is the number of cases in the knowledge base for this run; third, fourth and fifth columns are number of correct predictions for prediction 1, prediction 2 and prediction 3 respectively; and the last column shows how many predictions are outright misses. Figures 4.3 and 4.4 are charts for data in figure 4.2; Figure 4.5 is same as fig 4.3 with the difference that it is split in two parts and plotted on true scale. Figure 4.6 – 4.9 are corresponding figures for outcome field ‘Trouser Size’.

As mentioned earlier, we ran the test program in two variants: one which finds 5 nearest neighbors and second which finds 10 nearest neighbors. Since the results we obtained were similar, this chapter contains results only of predictions with 5 nearest neighbors. See appendix B for results of predictions with 10 nearest neighbors.

In general, we see a gradual learning in the system. The accuracy of prediction increases and the number of outright misses decreases as the number of cases in the knowledge base increases. The learning is more quick in the beginning but slows down as the knowledge base grows.

Figures 4.3 and 4.7 show several dips in the curves. Since the set of the test cases is the same throughout, and since additional cases were added to the knowledge base while keeping the old ones, one would not expect performance to ever drop. One hypothesis to explain this phenomenon is the inclusion of a batch of bad cases to the knowledge base. For instance we see in figure 4.3 the performance suddenly drops and the number of misses shoots up when the number of cases is 70. This suggests that a batch of bad cases might have been added to the knowledge base while increasing its size from 50 to 70.

In figure 4.3 and 4.6, the curves also show a lot of wiggling. This can be explained as follows: In most of the cases, the first, the second and the third predictions differ by a very small amount; for instance in short sleeve shirt size prediction, the three predictions may differ by only ± 0.5 sizes. Garment sizes in the cases were selected by different fitters. Each fitter's preference for 'ease' differs thus explaining some of the (small) variation in garment sizes for people having the same body measurements. Since each individual's preference in wearing slightly larger or smaller garment differs, a person might select the second or the third prediction even if the first prediction is a 'best fit'. Considering this, it would be wise to estimate the system performance by sum of two or three predictions instead of by just any one of them. The stacked charts in figures 4.4 and 4.8 show that the cumulative curves are smoother than their non-cumulative counterparts.

Another interesting observation is the relatively good performance of the system even at a very low number of cases. Figures 4.3 and 4.7 show that we have only 30% misses even when the knowledge base has a strength of just 5 cases. This could be very domain-specific. The outcome in Short Sleeve Shirt Size has a range of 12.5 – 18.5 (in increments of 0.5). Within this range, a majority of the cases lie between 13.5 – 16.0 (average person). Since most of the test cases are for 'average people' and most of the cases stored in the library tend to be for average people (even when there are only a few cases), there is a good chance of getting a correct prediction.

We see from the table that number of hits for prediction 1 never went above 67. With a knowledge this big, one would expect a better performance from the system. However, a library is only as good as the data on which it is built. Presence of bad cases can seriously affect the system performance. In fact adding a few bad cases to a library can actually worsen the performance. There could be several factors like inaccurate measurements, variance in garment tolerances etc. contributing to the shortcomings in the cases. These limitations are discussed in detail in the next chapter.

Summary of predictions for Short Sleeve Shirt Size

Run #	# of Stored Cases	# Hits for Prediction 1	# Hits for Prediction 2	# Hits for Prediction 3	Outright Misses
1	5	46	23	0	31
2	10	46	32	13	9
3	20	53	26	14	7
4	25	57	23	11	9
5	30	56	29	7	8
6	40	57	29	8	6
7	50	55	30	6	9
8	70	56	26	2	16
9	100	60	31	0	9
10	150	58	28	4	10
11	200	58	29	4	9
12	300	57	31	4	8
13	500	57	29	3	11
14	900	63	25	3	9
15	1400	63	25	3	9
16	1900	67	19	5	9
17	2400	64	27	1	8
18	2900	66	24	3	7
19	3956	67	27	2	4

Figure 4.2

Learning Curve:
for short sleeve shirt size prediction

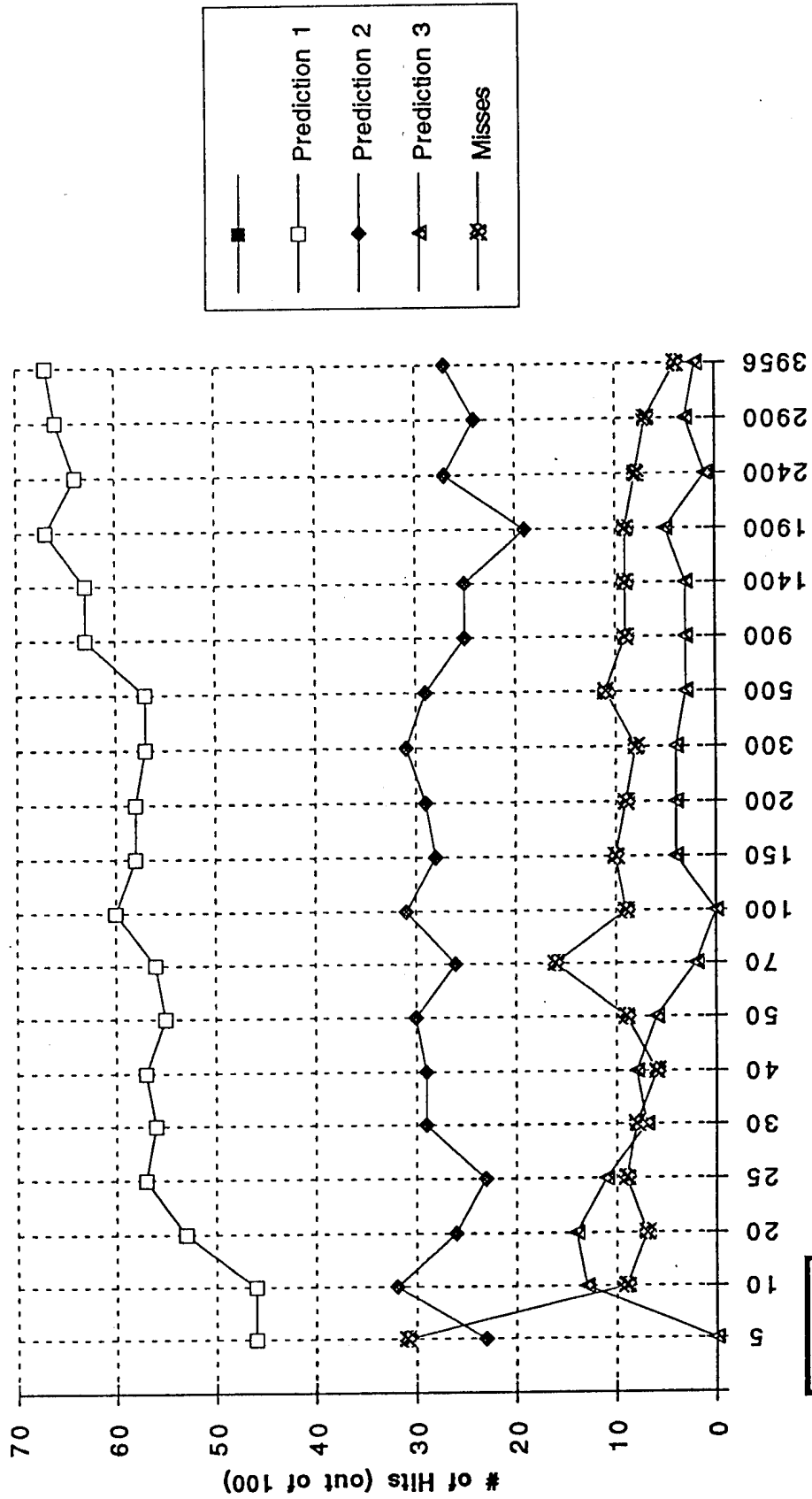


Figure 4.3

Learning Curve:
for short sleeve shirt size prediction
(Stacked Chart)

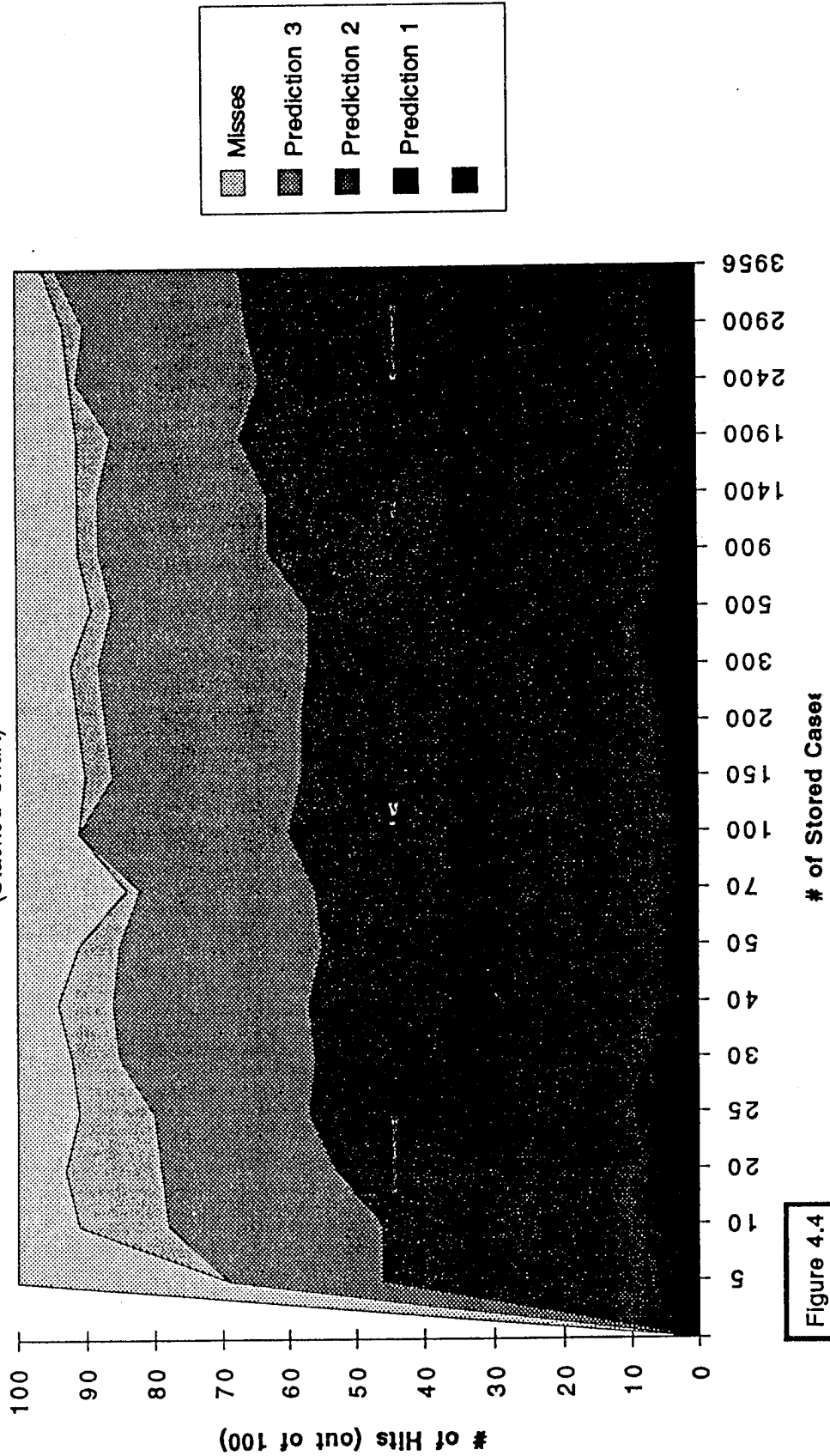


Figure 4.4

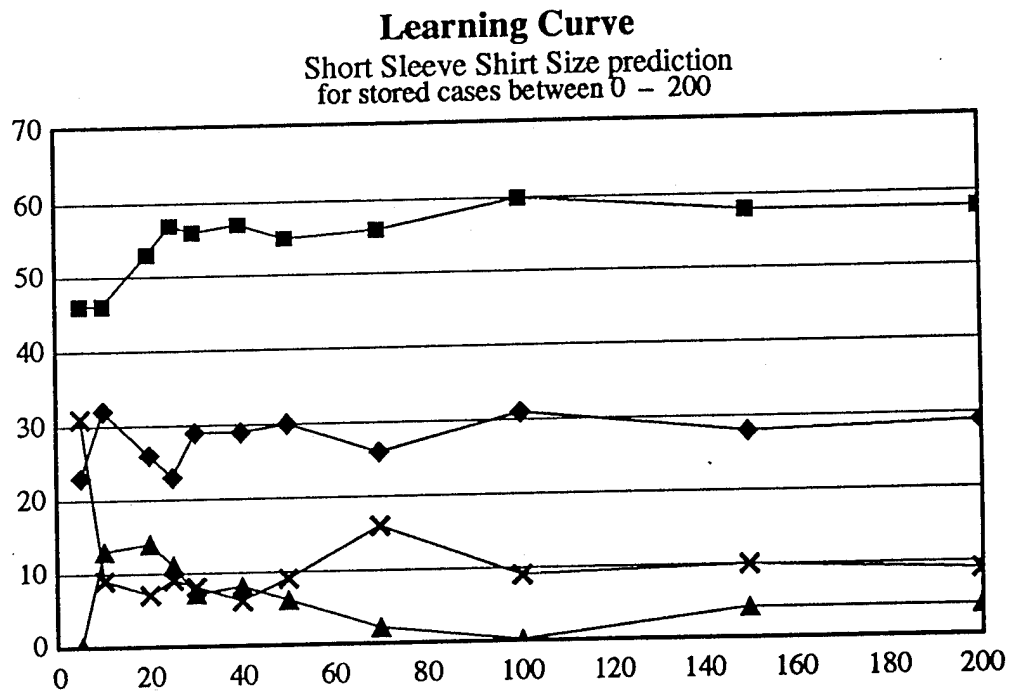


Figure 4.5(a)

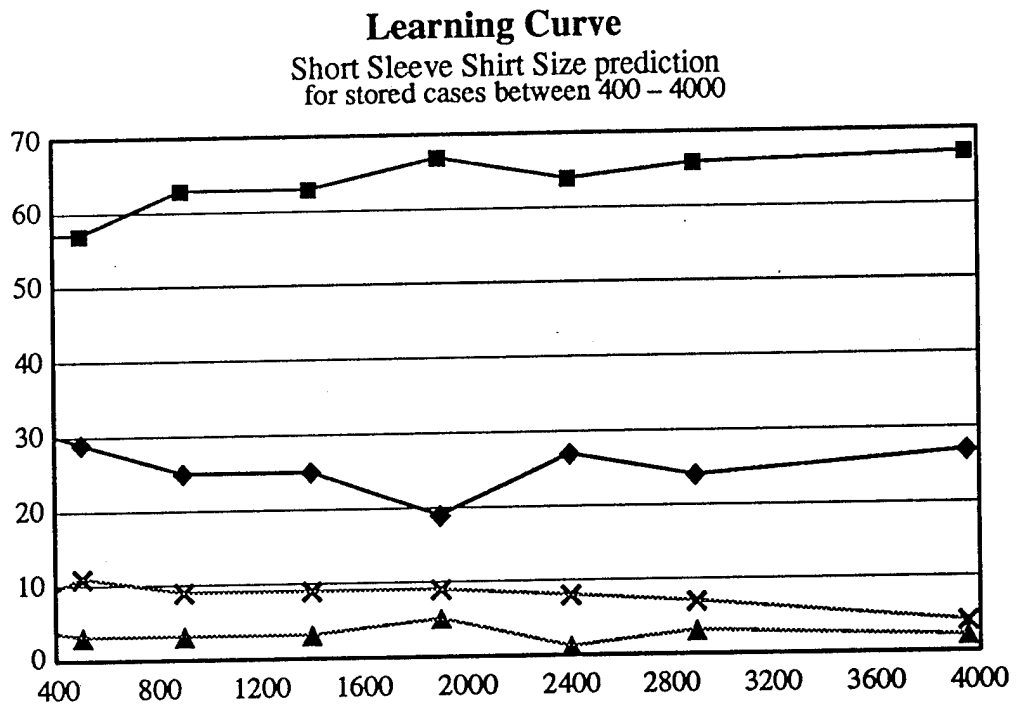


Figure 4.5(b)

Summary of predictions for Trouser Size

Run #	# of Stored Cases	# Hits for Prediction 1	# Hits for Prediction 2	# Hits for Prediction 3	Outright Misses
1	5	46	23	0	31
2	10	44	26	18	12
3	20	47	30	6	17
4	25	48	26	15	11
5	30	51	29	9	13
6	40	59	20	10	11
7	50	60	20	9	11
8	70	56	27	4	13
9	100	58	23	2	17
10	150	53	24	6	17
11	200	55	26	6	13
12	300	58	24	4	14
13	500	58	23	2	17
14	900	63	17	3	17
15	1400	57	24	3	16
16	1900	62	23	3	12
17	2400	56	26	4	14
18	2900	58	24	6	12
19	3956	57	29	5	9

Figure 4.6

Learning Curve:
for Trouser Size prediction

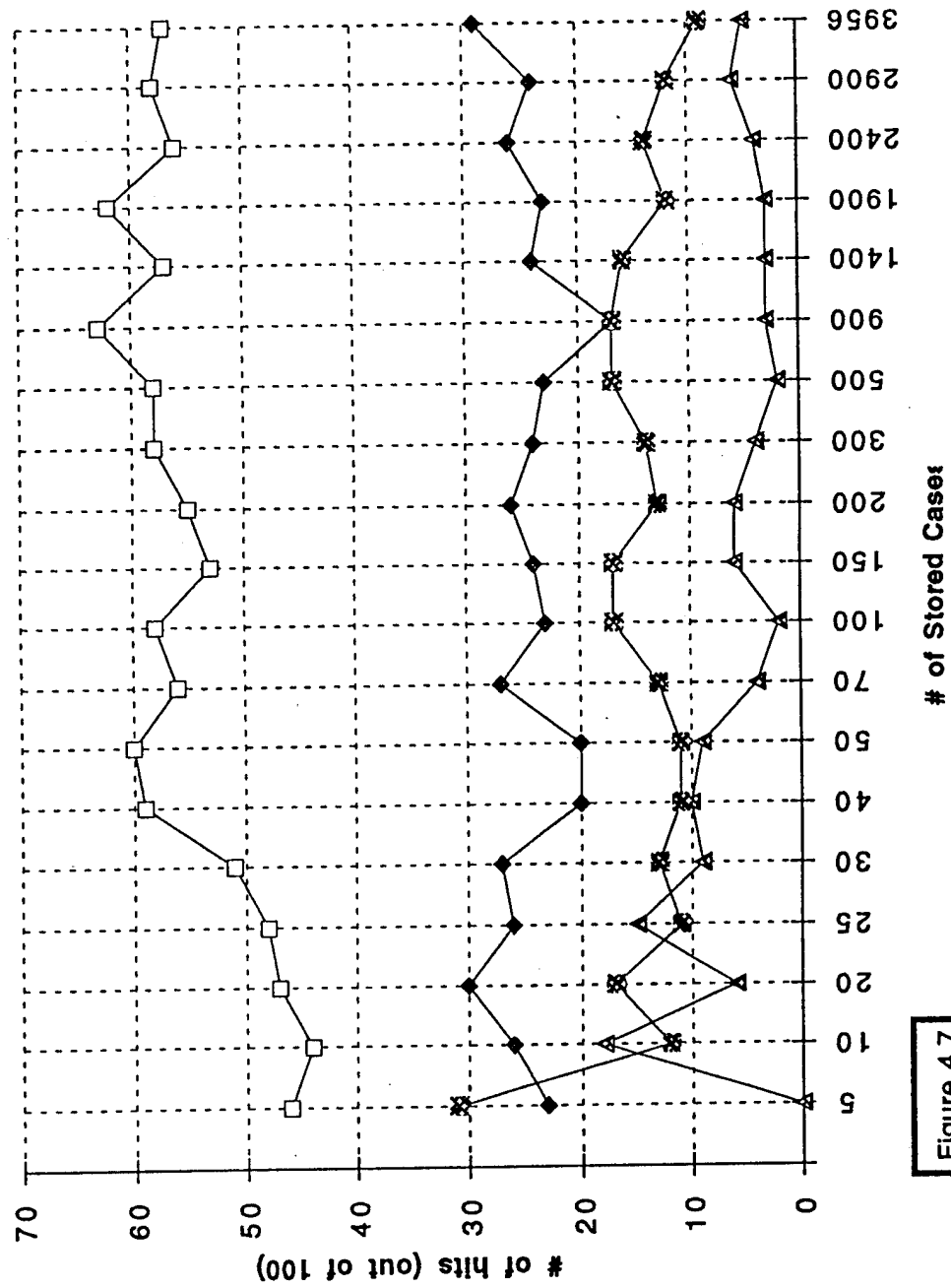


Figure 4.7

Learning Curve:
for Trouser Size prediction
(Stacked Chart)

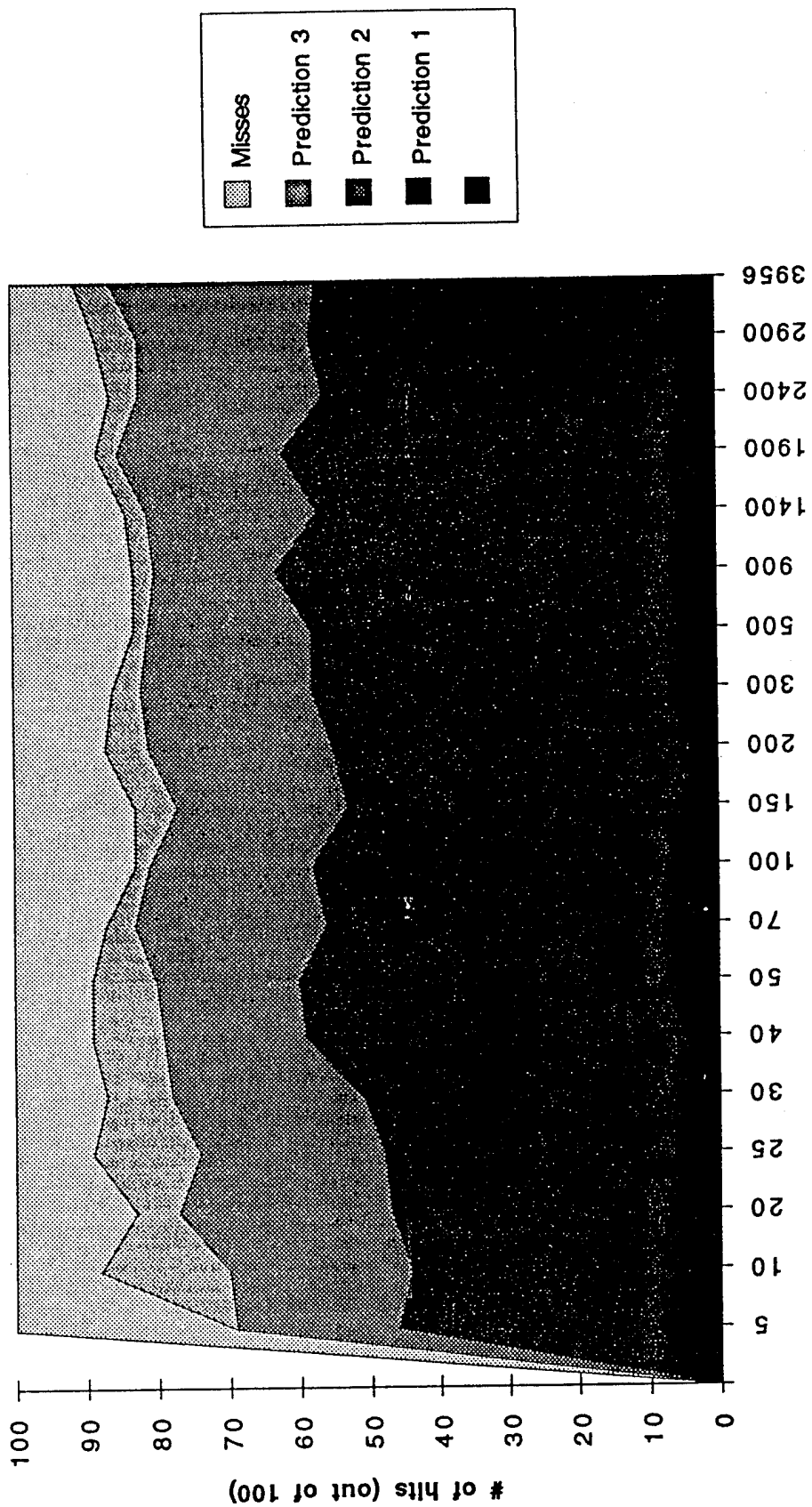


Figure 4.8

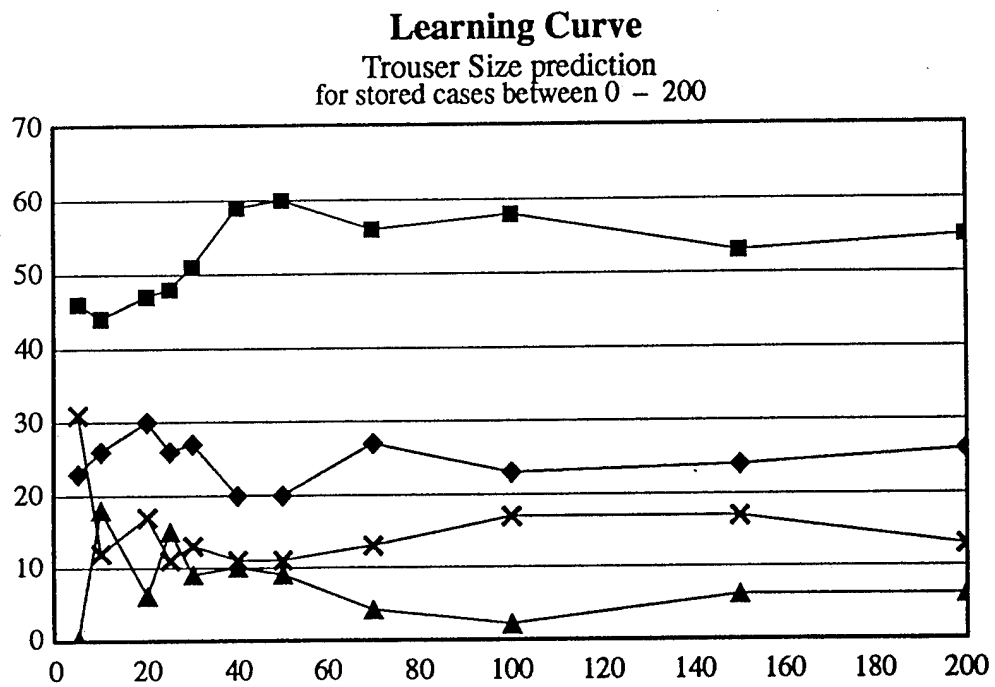


Figure 4.9(a)

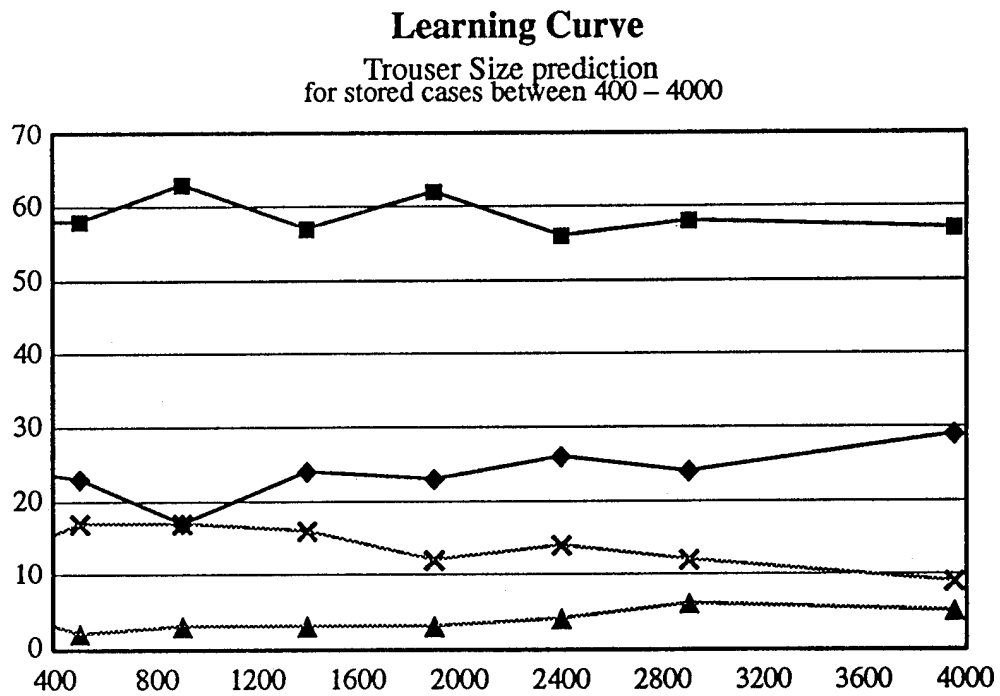


Figure 4.9(b)

Chapter 5

Conclusion

5.1 Limitations

The case base foundation of the system is sound but the system does not show a smooth learning curve. However, a library is only as good as the data on which it is built. Presence of 'bad cases' could seriously affect the system performance. Fitting is a very subjective field. Several factors contribute to the shortcomings in the cases used to build up the knowledge base.

1. Measurements are not always accurate. Sometimes soldiers with little training take the measurements. Professional measurers make mistakes because they operate under time pressure and get tired from handling several hundred soldiers in a few hours.
2. Measurements are not consistent. Data in the cases was produced by different professionals and soldiers who do not necessarily employ the same measurement approach.
3. The measurements in the cases are not a sufficient basis to determine garment sizes. Human fitters can take other factors into account when selecting sizes for try-on, but the expert system is limited to the data in the cases. The cases themselves suggest the insufficiency of the measurements. There are situations where a group of soldiers all have almost the same body measurements, but several different sizes of a garment were issued to soldiers in that group.
4. Variance in garment tolerances could cause inaccuracies. This variance can cause problems whether the size prediction is manual or automated. Given the assumption that out-of-tolerance garments are exceptional, the cases stored in the expert system should be based only on issues of in-tolerance garments. Therefore garment tolerance should be manually checked when cases are gathered to calibrate the expert system, to insure data in the cases is valid.

The system might actually be performing better than what it appears to be. In the research reported here, the system performance is compared to that of a human expert who works under time pressure and monotony. The fitter does not necessarily issue a "best fit" to the soldier. So even if the system predicts a best fit, its prediction may not match with the expert's selection and therefore in this research it would be considered wrong.

5.2 Lessons Learned

The project turned out some very interesting and unexpected results. There were many lessons learned as we worked on this project. I gained insight on working of case based reasoning systems in general and on the ReMind expert shell specifically. Programming the test procedure helped me learn the runtime interfacing with the expert system shell using the ReMind API (Application Program Interface). A more challenging experience was analyzing the data, interpreting the graphs and explaining the system behavior. Overall, I got a feel of the intricacies involved in developing an expert system and comparing a computer's performance to that of a human expert.

5.3 Future Work

This work can be considered as a first step towards the study of learning curve. There are a number of enhancements that could have improved the learning and performance of the system, but were not addressed due to the time constraints. Some of them are :

- Identify and eliminate the "bad cases" from the knowledge base. This could be done by predicting outcomes for each case using rest of the library. Another clue to identify bad cases would be checking those cases that are added just before the prediction curves dip. We can expect to find some bad cases in that batch.
- The study could be continued with an even larger knowledge base to see if the performance levels off at a certain point.
- Preparing several different sets of test cases and repeating the study with each of them might help eliminate the effects of any biasing within the test cases.
- Repeating the test for some other outcome field might make a difference. Selection of a field which is not so subjective, for instance prediction for cap size, might actually improve the learning curves.
- Body measurements considered may not be sufficient to predict the garment size. Some additional knowledge may be required possibly in form of added fields in the cases.
- One could try different combination of weight vectors to find out what works the best.

Appendix A

Test Program Source Code

```

/*****
**   File Name      : nnr.cpp
**   Comments       : This file contains the complete program for
**                   testing the learning curve of a knowledge base.
**
**   Date Created   : May 02, 1994 — Written by Vinit Jindal
**   Date Modified  : May 06, 1994 — Documented by Vinit Jindal
**
*****/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "CBReMind.h"

#define N_N_NUM 5

/* Function required by the qsort routine
*/
int comparevotes(const void *, const void *);

/* This structure is used in Voting
*/
typedef struct VoteStructTag {
    int numVotes;
    char valVote[16];
}VoteStruct;
```



```

/* Global Variables
*/

VoteStruct ballots[N_N_NUM];
int n_n_num = N_N_NUM;

/* Main Program
*/

int main (int argc, char *argv[] )
{
    CBRErrorInfo * err = 0;          /* Pointer to ReMind error Struct*/

    PCHAR    libName,                /* ReMind's equivalent of string */
             ViewName,
             WeightVectorName,
             TemplName;

    LCOUNT NumberOfCases;          /* ReMind's equivalent of Integer
*/

    /* Handles to ReMind objects */
    CBRLibHandle    theLib;
    CBRViewHandle   theView;
    CBRWVHandle     theWV;
    CBRFieldId      theFld;
    CBRTmplHandle   theTemplate;

    CBRCaseList toSearchCases,       /* Pointers to different case lists */
             hypoCases,
             outCases,
             allCases;

    CBRScore * outScores;            /* Pointers to scores lists */
    CBRScore * outScores_orig;

    CBRCaseId  outCaseId,
             hypoCaseId;

    FILE * out_file;                /* Results file */

    CBRValue  *theValue;
    char      *value_str;

    /* Templates names used */
    char template_str[][32]=
    {"100to105"}, {"100to110"}, {"100to120"}, {"100to125"}, {"100to130"},
    {"100to140"}, {"100to150"}, {"100to170"}, {"100to200"}, {"100to250"},
    {"100to300"}, {"100to400"}, {"100to600"}, {"100to1000"}, {"100to1500"},
    {"100to2000"}, {"100to2500"}, {"100to3000"}, {"100to4056"}
};

```

```

/* File name for results */
char file_name_str[32];

int j, run_count;

if ( argc != 4 ) {
    printf ("Unmatched number of arguments");
    return 1;
}

libName = argv[1];
printf ("\nLibrary is %s", libName);

ViewName = argv[2];
printf ("\nView is %s", ViewName);

WeightVectorName = argv[3];
printf ("\nWeight Vector is %s", WeightVectorName);

NumberOfCases = N_N_NUM;

printf ("\nStarting the API");

/* Start the API */
if (!CBRStart (0) ) {
    err = CBRGetError();
    printf ("Start Error. Severity=%d; Code=%d; Text=%s\n",
        err->severity, err->code, err->text );
    return 1;
}
else printf ("\nAPI Started");

/* open the Library */
if (!CBROpen (libName, CBR_OPEN_NO_BACKUP, &theLib) ) {
    err = CBRGetError();
    printf ("Open Error. Severity=%d; Code=%d; Text=%s\n",
        err->severity, err->code, err->text );
    return 1;
}

/* Get the View */
if (!CBRGetView (theLib, ViewName, &theView) ) {
    err = CBRGetError();
    printf ("View Error. Severity=%d; Code=%d; Text=%s\n",
        err->severity, err->code, err->text );
    return 1;
}

```

```

/* Get the WeightVector */
if (!CBRGetWV (theView, WeightVectorName, &theWV )) {
    err = CBRGetError();
    printf ("WV Error. Severity=%d; Code=%d; Text=%s\n",
            err->severity, err->code, err->text );
    return 1;
}

/* Make List of all cases in the library */
if (!CBRMakeCaseListByDisposition (theView, CBR_ANY_DISP, &allCases ))
{
    err = CBRGetError();
    printf ("All Case List Error. Severity=%d; Code=%d; Text=%s\n",
            err->severity, err->code, err->text );
    return 1;
}

/* Make Hypo cases List.
   This would be list of test cases.
   */
if (!CBRMakeCaseListByDisposition (theView, CBR_HYPOTHETICAL,
                                   &hypoCases ))
{
    err = CBRGetError();
    printf ("Hypo Case List Error. Severity=%d; Code=%d; Text=%s\n",
            err->severity, err->code, err->text );
    return 1;
}

/* Get field handle */
if (!CBRGetField(theView, "Short Sleeve Shirt Size", &theFld))
{
    err = CBRGetError();
    printf ("Field Error. Severity=%d; Code=%d; Text=%s\n",
            err->severity, err->code, err->text );
    return 1;
}

run_count = 0;

/* Set first template as current template
   */
TemplName = template_str[run_count];

```

```

/* Repeat until all the templates are done
*/
while (CBRGetTpl (theView, TmplName, &theTemplate))
{
    sprintf (file_name_str, "outfil%d.txt", run_count);
    if ((out_file=fopen (file_name_str, "w"))==NULL) {
        printf ("\nUnable to open file %s\nQuitting....\n",
            file_name_str);
        exit (-1);
    }

    fprintf(out_file, "Results for View : %s\n", ViewName);
    fprintf(out_file, "Template used : %s\n", TmplName);
    fprintf(out_file, "Nearest Neighbour : %d\n", n_n_num);

    printf ("\n\nMaking CaseList by Template #%d - %s.....",
        run_count+1, TmplName);

    /* Prepare list of cases to search
    using the current template.
    */
    if (!CBRTmplRetrieve (theView, theTemplate, allCases,
        &toSearchCases))
    {
        err = CBRGetError();
        printf ("Template List Error");
        printf ("\Severity=%d; Code=%d; Text=%s\n",
            err->severity, err->code, err->text );
        return 1;
    }

    printf ("Done !!");

    /* Perform NNR for each of the test
    cases using the search case list
    */
    printf ("\nPerforming NNR for case : \t");
    CBRRestartCaseList ( hypoCases );

    while(CBRGetNextCaseId(hypoCases,&hypoCaseId))
    {
        printf ("\b\b\b\b\b\t%d", (long)hypoCaseId);

        /* Perform a nearest neighbour retrieval */
        if (!CBRNNRetrieve (theView, theWV, hypoCaseId,
            toSearchCases, NumberofCases, &outCases,
            &outScores))
        {
            err = CBRGetError();
            printf ("NNR List Error");
            printf ("\Severity=%d; Code=%d; Text=%s\n",
                err->severity, err->code, err->text );
        }
    }
}

```

```

        return 1;
    }
    else
        outScores_orig=outScores;

        /****** Collect the results *****/

        /* Empty ballot box */
        for(j=0;j<N_N_NUM;++j){
            ballots[j].numVotes = 0;
            strcpy(ballots[j].valVote,"n/a");
        }

        /* Vote on them
        */
        while(CBRGetNextCaseId(outCases,&outCaseId)) {
            if(!CBRGetFldValue(theView,outCaseId,theFld,
                               &theValue))
            {
                printf ("Unable to get field value\n");
                return 0;
            }

            if(!CBRFormatFldValue(theView,theFld,theValue,
                                   &value_str))
            {
                printf ("Unable to format field value\n");
                return 0;
            }

            j=0;

            while((j<N_N_NUM)&&(strcmp(ballots[j].valVote,
                                       "n/a")!=0))
            {
                if(strcmp(ballots[j].valVote,value_str)==0) {
                    ballots[j].numVotes++;
                    break;
                }
                else j++;
            }
            if(strcmp(ballots[j].valVote,"n/a")==0){
                strcpy(ballots[j].valVote,value_str);
                ballots[j].numVotes = 1;
            }
            CBRFree ( theValue );
            CBRFree ( value_str );
        }

```

```

/* Sort the ballots
*/
qsort(ballots,N_N_NUM,sizeof(struct VoteStructTag),
      comparevotes);

CBRGetFldValue(theView,hypoCaseId,theFld,&theValue);
CBRFormatFldValue(theView,theFld,theValue, &value_str);

/* Print Case ID, Original Size, and
three predicted choices
*/
fprintf(out_file,
        "%ld, %6s, %6s,%d, %6s,%d, %6s,%d\n",
        hypoCaseId, value_str,
        ballots[0].valVote, ballots[0].numVotes,
        ballots[1].valVote, ballots[1].numVotes,
        ballots[2].valVote, ballots[2].numVotes
);

/* Free the memory
*/
CBRFree(theValue);
CBRFree(value_str);
CBRFree ( outCases );
CBRFree ( outScores_orig );
}

printf ("\nDone !!");
fclose (out_file);
CBRFree ( toSearchCases );

/* Select next template
*/
run_count++;
TemplName = template_str[run_count];
}

CBRFree ( hypoCases );
CBRFree ( allCases );

/* Close the library
*/
if ( !CBRClose ( theLib, CBR_CLOSE_COMMIT, CBR_REMOVE_BACKUP ))
{
    err = CBRGetError();
    printf ("Close Error\nSeverity=%d; Code=%d; Text=%s\n",
           err->severity, err->code, err->text );
    return 1;
}

```

```

/* Shut Down API
*/

if ( !CBREnd(0) ) {
    err = CBRGetError();
    printf ("End Error\\Severity=%d; Code=%d; Text=%s\\n",
        err->severity, err->code, err->text );

    return 1;
}

return 0;
}

```

```

/*
**
** Function required by the qsort routine
**
*/
int comparevotes(const void *one, const void *two)
{
    VoteStruct *a,*b;

    a = (VoteStruct *) one;
    b = (VoteStruct *) two;

    if ( a->numVotes < b->numVotes) return 1;
    else if( a->numVotes > b->numVotes) return -1;
    else return 0;
}

```


Appendix B

Test results with 10 Nearest Neighbor Search

**Summary of predictions for
Short Sleeve Shirt Size
With 10 Nearest Neighbor Search**

Run #	# of Stored Cases	# Hits for Prediction 1	# Hits for Prediction 2	# Hits for Prediction 3	Outright Misses
1	25	59	23	0	18
2	50	56	0	0	44
3	100	58	28	8	6
4	150	64	21	7	8
5	200	58	27	7	8
6	300	57	36	2	5
7	500	55	35	6	4
8	900	59	31	6	4
9	1400	61	31	4	4
10	1900	63	26	7	4
11	2400	67	26	4	3
12	2900	66	25	6	3
13	3956	71	25	3	1

Figure B-1

Learning Curve:
Short Sleeve Shirt Size Prediction with 10 Nearest Neighbor Search

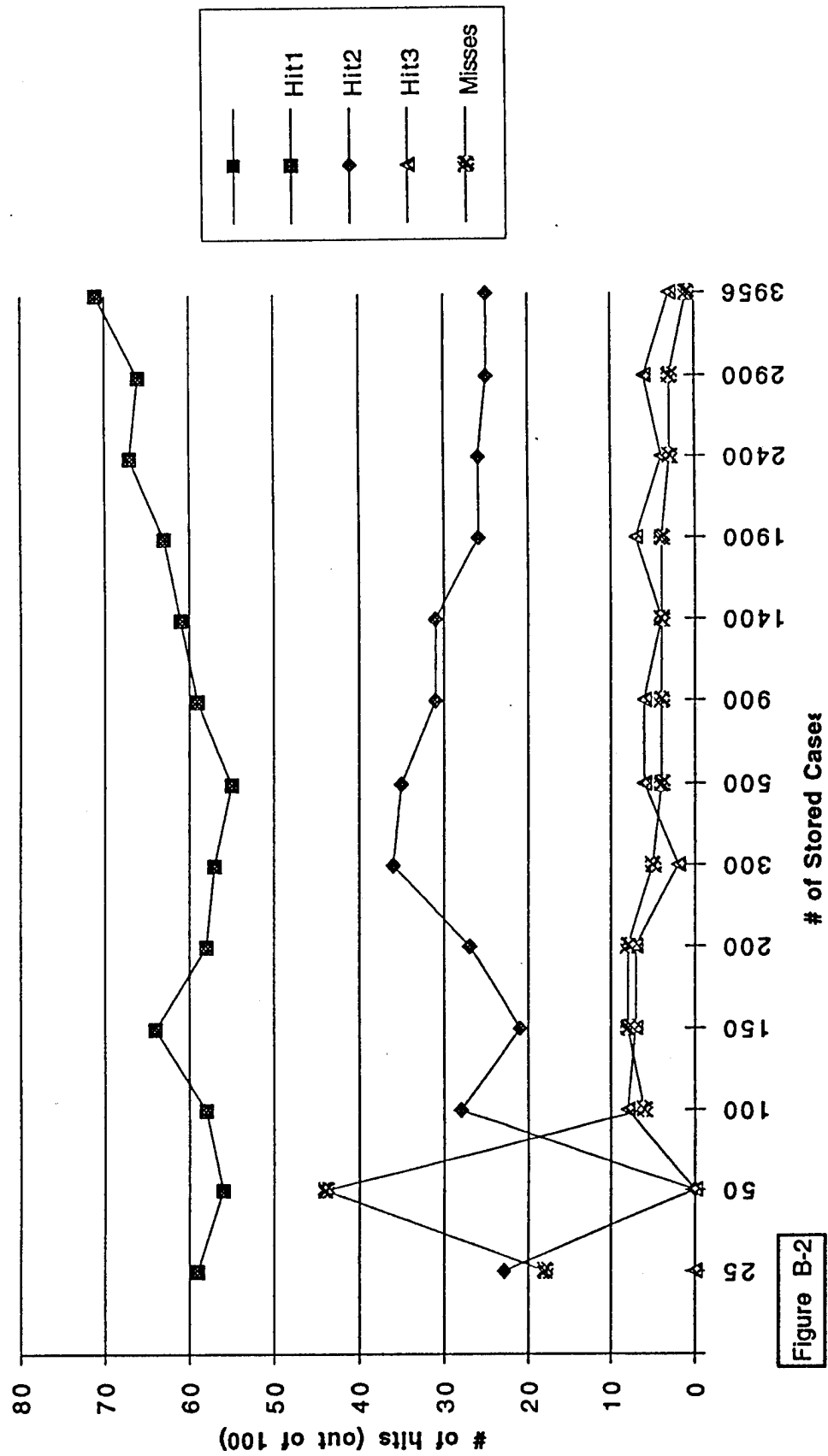


Figure B-2

Learning Curve:
Short Sleeve Shirt Size Prediction with 10 Nearest Neighbor Search
(Stacked Chart)

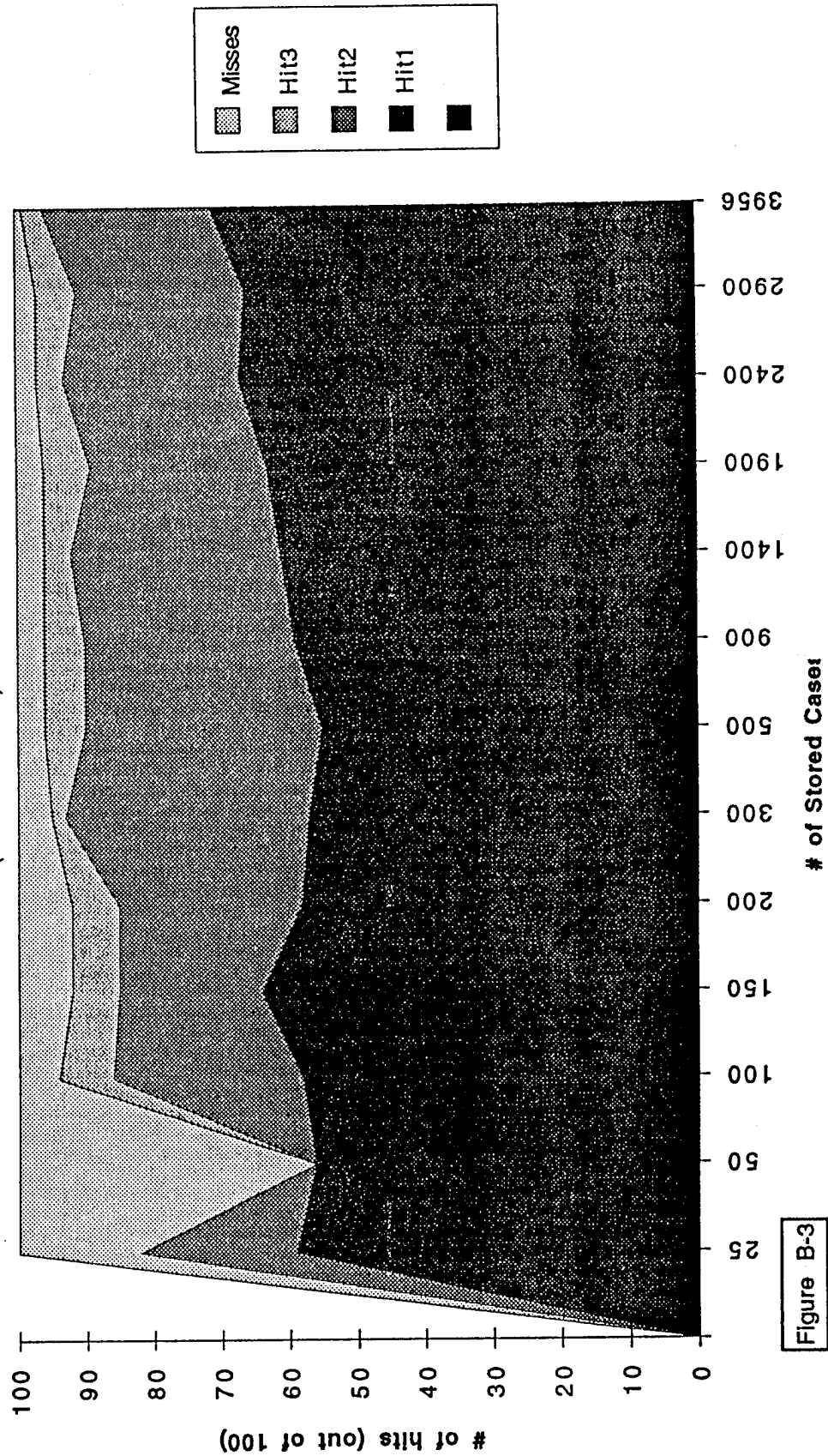


Figure B-3

Bibliography

- [1] J. A. Hartigan. *Clustering Algorithms*. John Willey and Sons, 1975.
- [2] Ray Bareiss. *The Experimental Evaluation of a Case-Based Learning Apprentice*. Case-Based Reasoning Workshop, May 1989.
- [3] Agnar Aamodt. *Towards Expert Systems that Learn from Experience*. Case-Based Reasoning Workshop, May 1989.
- [4] Cognitive Systems, Inc. *ReMind Developer's Reference Manual*. Cognitive Systems, Inc. 1992.
- [5] Cognitive Systems, Inc. *ReMind Developer's API Draft*. Cognitive Systems, Inc. 1992.
- [6] Hennessy D, Hinkle D, *Applying Case-Based Reasoning to Autoclave Loading*. IEEE Expert, October 1992, IEEE Press.
- [7] Case-Based Reasoning, *Proceedings of a Workshop on Case-Based Reasoning*, Pensacola Beach FL, May 1989, Morgan Kaufmann Publishers, Inc.
- [8] Kolodner J and Riesbeck C. *Experience, Memory and Reasoning*. Lawrence Erlbaum Associates, Publishers. 1986.
- [9] Simoudis Evangelos. *Using Case-Based Retrieval for Customer Technical Support*, IEEE Expert, October 1992, IEEE Press.
- [10] Borland Inc. *Borland C++ 3.1 Programmers Reference Manual*.

Appendix C

3DM documentation

3DM software documentation

1.0 Overview

This section describes the software developed in this study. Called 3DM, the software is written in C and runs on a SUN workstation. The code utilizes X-window graphics libraries to manage the windowing environment. The measurement language code, in particular, continues to be developed.

2.0 User Interface

When started, 3DM activates five windows: (a) Menu, (b) Figure Display, (c) Image Name, (d) Measurements, and (e) Views. A digitized image is automatically selected and displayed in the Figure Display window.

MENU allows the user to rotate, translate, and scale the image. The user may rotate the image along X-, Y-, or Z-axes after setting the angle of rotation. Translation is also along the X-, Y-, or Z-axes and the user may set the translation distance. The user may scale the image up or down. Any of these operations may be combined allowing the user to view the figure from practically any perspective. With AUTO ROTATE, the user may cause the figure to rotate along one of the three axes by the angle specified for the axis, pause briefly, and rotate again. Random rotation allows the figure to rotate in any or all of the three axes.

2.1 LOAD

To load an image, the user clicks on the Load button and makes a selection from among the images available (IMAGES window, Figure 3). The name of the image file is displayed in the IMAGE NAME window. The IMAGES window is dismissed by clicking on Done.

2.2 MEASUREMENTS

The MEASUREMENTS windows, Figure 4a, provide the user with three types of measurements and a macro language facility. The first window provides the options, the second displays the results.

2.2.1 Multipoint

Multipoint measurement allows the user to take the total surface distance between two or more points selected on the figure. This provides the user with a simulation of a tape measurement on the figure. The measurement is displayed in inches in the second MEASUREMENTS window. When Multipoint measurement is active, the mouse buttons function as follows:

Multipoint

- Left Button—select a point on the figure
- Middle Button—de-select most recently selected point
- Right Button—measure surface distance between selected points

2.2.2. Radial

Radial measurement allows the user to take the surface distances between one source point and one or more destination points selected on the figure. Unlike

Multipoint, Radial measurement returns several distances, one for each destination point selected. When Radial measurement is active, the mouse buttons function as follows:

Multipoint

- Left Button—select source or destination points on the figure
- Middle Button—de-select most recently selected destination point
- Right Button—measure surface distances between selected points

Multipoint or Radial measurements are calculated through the use of Dijkstra's shortest path algorithm on a graph imposed on the set of points on the figure. This algorithm and graph are described in detail in Section 4.

2.2.3. Circumference

Circumference provides a measurement around the figure's torso, arms, or legs (Figure 4e). The user clicks on any point on the figure and a horizontal slice of the figure through the point is taken and measured. The measurement, in inches, is displayed in the MEASUREMENTS window. A summary of the mouse button functions is given in Figure 4e.

Multipoint

- Left Button—select and measure horizontal slice on the figure
- Middle Button—not used
- Right Button—end Circumference selection

Figure 4d

2.2.4. Language

A simple macro language is provided to provide the user with the capability to develop small macros, i.e., sequences of commands, for the purpose of automating the process of measurement taking. For example, a macro called chest may be developed to locate and measure the figure's chest automatically. Similarly, macros may be developed to measure waist, seat, and sleeve length. This language is described in detail in Section 6.

2.3. SLICES

The user may obtain slices of the figure along the XY, XZ, and YZ planes. The user's slice view options are: Vertical Y, Vertical Z, and Collapsed View. These options provide, for example, views of the posture, slope of the shoulders, or waistline of the figure. In Figure 5 the user has selected the Vertical Y view and can observe the curvature of the figures front and back. Figure 6 shows a collapsed view of the waist and seat of the figure as seen from the top. The option of viewing slices of the figure allows the user to isolate selected features of the figure for closer scrutiny.

2.4. RESET, QUIT

The last two buttons on the MENU are RESET which returns the figure to its original position, and QUIT which terminates 3DM.

3.0 3DM Language

A summary of the commands available in the 3DM language is shown below. The general format of a command is:

command option

where a command is one of five types: Region, Slice, Point, Measurement, and Control. Options vary with each command.

A Region command selects an region of the figure that the user wants to study further and measure. The regions that may be selected are the torso, left arm, right arm, left leg, right leg, upper body, and the entire figure. This enables the user to focus on a particular part of the figure which makes it easier to isolate certain slices or points. Note that one of the options is the entire figure. A region must be selected before the user issues Slice or Point commands.

The user may enter a Slice command to select a specific slice within the region selected. Similarly, Point commands allow the user to select a point within a slice. Slices and points may be saved and used with other Slice, Point, and Measurement commands. There are several Slice commands available: selectslice, topslice, middleslice, and bottomslic. Selectslice allows the user to select a slice numerically, i.e., as a percentage of the region selected. Topslice, middleslice, and bottomslic are included for convenience, and are equivalent to selectslice 0, selectslice 0.5, and selectslice 1, respectively.

After selecting a slice, the user may modify the selection through the slicemove command which changes the focus from the current slice to another. The slicemove command requires a direction, (UP, DOWN) and a distance expressed in inches.

The current slice may be saved for future use using the slicesave command, which saves the current slice and assigns to it a name. This name may be used in other Slice commands (see maxslice and minslic below). A named slice is used with commands currentslice, minslic, and maxslice. Currentslice selects a previously named slice. Minslic (maxslice) selects the slice with the smallest (largest) circumference between two named slices.

Point commands allow a user to select individual points within a slice. Two commands available are the center and most commands. Center takes a position (FRONT, BACK, LEFT, RIGHT) and selects the point closest to the center-front, center-back, center-left, center-right of the current slice. Similarly Most takes a position and selects the frontmost, backmost, leftmost or rightmost point of the current slice. A selected point may be changed with the pointmove command. After a point has been selected, it may be saved with the saveto command. A previously saved point may be recalled with currentpoint :

The system is designed so that the user systematically focuses on slices and points. He or she first selects a region of interest, selects slices within the region, selects points within slices, saving those slices and points which are of interest. At this stage, the user is ready to take measurements.

There are one slice measurement function (circum) and two point measurement functions (directdist, surfacedist) available. Circum returns the circumference, in inches, of a selected slice. Directdist takes a list of points, P1 P2...Pn, and returns the direct, Euclidean, distance between consecutive points P1 and P2, P2 and P3, etc. Surfacedist is similar to directdist. It takes the surface distance between consecutive points P1 and P2, P2 and P3, etc.

Three control functions are available: runfiles, help, and quit. Runfiles allows a user to develop and execute a macro, i.e., a sequence of commands within a file. The user must use an editor to create the file. This makes it convenient, for example, for the user to take a chest measurement of a figure. For example, the user may enter the following commands into a filename named chest measurement,

1. chest measurement:
2. region torso
3. slicemove down 1.0
4. slicesave chest
5. circumference chest

and will simply have to key: Runfiles Chest Measurement to take the chest measurement of any figure currently loaded. Help provides a list of commands accepted by 3DM and Quit terminates the language interpreter.

3DM Language Commands

Region area

torso
larm
rarm
lleg
rleg
upbody
all

Slice

selectslice percent

Percent is an integer between 0 and 100. It allows the user to specify a slice some percentage distance from the top of the region, where the distance from top to bottom represents 100. Selectslice 10, for example, will select the slice approximately 10 percent from the top of the region.

topslice

Select the top slice of the selected region; equivalent to selectslice 0.

middleslice

Select the middle slice of the selected region; equivalent to selectslice 50.

bottomslice

Select the bottom slice of the selected region; equivalent to selectslice 100.

slicemove direction distance

Direction is either UP or DOWN and distance is a positive numeric value representing a distance in inches. For example, slicemove UP 1.5 moves up

from the current slice a distance of 1.5 inches. The newly selected slice becomes the current slice.

slicesave name

Save the current slice and assigns it a name. This name may be used in other Slice commands (see maxslice and minslice below). Name may be any string, starting with a non-blank character and continuing with any character (including blanks and punctuation) up to a maximum length of twenty characters. The following are all valid names: mid-thigh , abdominal area, slice 12, KNEE.CAP .

currentslice name

Select a previously saved slice. The selected slice may be used further in other commands (see maxslice, minslice).

maxslice slice1 slice2

Slice1 and slice are slice names (see slicesave). The command maxslice selects from the current region the slice with the largest circumference between slice1 and slice2. Slice1 must be above Slice2. In the case of a tie, the slice closest to slice1 is selected.

minslice slice1 slice2

Slice1 and slice2 are slice names (see slicesave). The command minslice selects from the current region the slice with the smallest circumference between slice1 and slice2. Slice1 must be above Slice2. In the case of a tie, the slice closest to slice1 is selected.

Point

center position

Position is one of: front, back, left, right. This command selects the point closest to the center-front, center-back, center-left, center-right of the current slice.

most position

Position is one of: front, back, left, right. This command selects the frontmost, backmost, eftmost or rightmost point of the current slice.

pointmove direction inches

Direction is one of: up, down, clock, counterclock and inches is the distance of the move expressed in inches.

saveto name

Save the current point and assigns to it a name. This name may be used in other commands (see directdist, surfacedist). Names for points follow the same rules for slice names (see slicesave).

currentpoint name

Select a previously saved point.

Measurement

circum

Return the circumference, in inches, of the most recently selected slice.

directdist P1 P2... Pn

Take the direct, i.e., Euclidean, distance between consecutive points P1 and P2, P2 and P3, etc.

surfacedist P1 P2... Pn

Take the surface distance between consecutive points P1 and P2, P2 and P3, etc.

Control

runfiles filename

Executes the sequence of instructions contained in filename. The file should be written using a text editor and should contain one 3DM language command per line.

help

Displays list of commands.

quit

Terminates the interactive session.

Appendix D

3DM source code

**Source Code for
3DM: Software for the DLA
3D Noncontact Measurement Project**

**Roy P. Pargas
Shan Jiang
Jasbir Manotra**

*Clemson Apparel Research
500 Lebanon Road, Pendelton, SC 29570
tel: 803-646-8454
fax: 803-646-8230
email: pargas@cs.clemson.edu*

```
# include "common.h"
# include <X11/Xlib.h>
# include <X11/Xutil.h>
```

```
extern Display *ncdisplay;
extern Window ncwindow;
```

```
extern int numSegs,numCurves;
```

common.h Thu Feb 24 14:21:03 1994

```
# ifndef _COMMON_H
# define _COMMON_H

# define PI 3.1415926535897932384626

# define SUN_VERSION 1

# define MAX_DFILES 10

/* max no. of segments in a sequence of clicks */
# define MAX_SEGMENTS 20

# define MAX_POINT_NUM 10000

# define X_AXIS 1
# define Y_AXIS 2
# define Z_AXIS 3
# define RANDOM 4
# define NO_AUTO_ROTATE 0

# define INCREASE 1
# define DECREASE 0

/* 3D point data structure */
struct pt3 {
    double x;
    double y;
    double z;
};

typedef struct pt3 point3;

/* 2D point data structure */
struct pt2 {
    double x;
    double y;
};

typedef struct pt2 point2;

typedef struct curve_dis_st {
    int src, snk;
    int length;
    double ds;
    int path[1000];
} curve_dis;

# endif
```



```
# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include <stdio.h>

# define DF_FG_RED 0
# define DF_FG_BLUE 60000
# define DF_FG_GREEN 60000

# define DF_BG_RED 30000
# define DF_BG_BLUE 30000
# define DF_BG_GREEN 30000

# define FN_WIDTH 25
# define MAX_DFILES 20
# define MAX_FILE_NAME 50
# define STR_LOC_X 20
# define STR_LOC_Y 15

extern Display *ndisplay;
extern Window menu_window, ncwindow, meas_win, cur_fname_win;
extern GC cur_fname_gc;
extern XEvent ncevent;
extern long hbg, hfg;
extern int ncscreen;
extern Colormap nc_cmap;
```

```

#include "disp.h"

#define SUCCESS 1
#define FAIL 0
#define LIMIT 100

extern point3 *showPoints;
extern int numPoints;
extern int allSlices[MAX_POINT_NUM];
extern int numSlices;
extern double distance();
extern int adjacent[2][MAX_POINT_NUM];

struct sp_tree_st {
    int num; /* point entry in allPoints */
    double wgt; /* weight from source */
    int pred; /* predecessor */
    int count; /* comparison count, used to speed up alg. */
};

typedef struct sp_tree_st SPAN_TREE_ST;

SPAN_TREE_ST span_tree[MAX_POINT_NUM]; /* nodes whose shortest path are known */

/* Dijkstra algorithm using heap sort */
SPAN_TREE_ST span_temp[MAX_POINT_NUM]; /* temporary storage of spanned nodes */
int temp_last;
int part_tree[MAX_POINT_NUM]; /* serves as index to span_temp */
int part_last;
int mark_temp[MAX_POINT_NUM]; /* if a node is not in temp, it is marked -1;
                                else it is assigned the entry in part_tree. */

int marked[MAX_POINT_NUM]; /* if a node is not spanned, it is marked -1;
                              else it is assigned the entry in span_tree. */

/*typedef struct curve_dis_st {
    int src, snk;
    int length;
    double ds;
    int path[1000]
} curve_dis;
*/

```

```

# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include <stdio.h>
# include <math.h>
# include <string.h>

# include "common.h"

# define MAX_POINT_NUM 10000
# define X_AXIS 1
# define Y_AXIS 2
# define Z_AXIS 3
# define INCREASE 1
# define DECREASE 0

# ifdef SUN_VERSION
extern Display *ncdisplay;
extern Window ncwindow, menu_window, meas_win, df_win, fname_wins[], cur_fname_win;
extern Window measMenu_win, meas_wins[];
extern Window sliceMenu_win, slice_wins[];
extern GC ncgc, menu_gc, meas_gc;
extern XSizeHints nc_hints, menu_hint;
extern int nc_screen;
extern unsigned long hfg, hbg;
extern int fnameCnt, measCnt, sliceCnt;
# endif

extern point3 *allPoints;
extern int numSlices;
extern int chest, waist, seat;

extern int transUnit;
extern double rotateUnit;
extern double scaleUnit;

```

```
# include <stdio.h>
# include <X11/keysym.h>
```

```
# include "common.h"
# include "slice_view.h"
```

```
extern int  *showPoints;
extern int  inslice[MAX_POINT_NUM];
```

```
# include <stdio.h>
# include <fcntl.h>
# include <sys/file.h>
# include "common.h"
# define MAX_POINT_NUM 10000
# define BUFFER_SIZE 1024
```

init.h Sat Dec 4 15:24:12 1993

```
# include <math.h>
# include <stdio.h>
# include <X11/Xlib.h>

# include "common.h"

# define STD_SP_SIZE 800

extern point3 *allPoints,*showPoints,rotOrigin;
extern int numPoints,speedUnit;
extern short incFlag,autoFlag;
extern int transUnit;
extern double rotateUnit,scaleUnit;
extern point2 cop;
extern double COS30,spsize;
extern int numCurves,numSegs,pickSlc;
extern int is_vert_slice,is_colp_slice;
extern int pickPnt[];

# ifdef SUN_VERSION
extern XPoint *scr_coords;
# endif
```

```

#include <string.h>
#include <ctype.h>
#include <stdio.h>

#define STR_INT 0
#define STR_CMD 1
#define STR_ID 2

#define TYPE_INT 0
#define TYPE_DOUBLE 1
#define TYPE_STRING 2
#define TYPE_RETURN 3
#define TYPE_LAST_STRING 4

#define STATE_UNCHANGE 0
#define STATE_REGION 1
#define STATE_SLICE 2
#define STATE_POINT 3

#define NONE 0
#define INT 1
#define DOUBLE 2
#define INT_INT 3
#define INT_DOUBLE 4
#define STRING 5
#define STRING_STRING 6
#define MULTI_STRING 7

#define NUM_ENTRY 20
#define NUM_RESERVED 40
#define MAX_STRING_LENGTH 32
#define STACK_SIZE 10
#define MULT_POINT_NUM 8

#define L 0
#define R 1

#define TORSO 0
#define LARM 1
#define RARM 2
#define LLEG 3
#define RLEG 4
#define UPBODY 5
#define ALL 6

#define UP 0
#define DOWN 1
#define CLOCK 2
#define COUNTERCLOCK 3

#define FRONT 0
#define BACK 1
#define LEFT 2
#define RIGHT 3

#define CMD 0
#define CONST 1

#define INTERACTIVE 0
#define BATCH 1

/* Finite state automata */
struct fsa_st {
    int next_state;
    int parameter;
    int (*func)();
};

typedef struct fsa_st fs_automata;

fs_automata fsa[NUM_ENTRY];

/* Reserved words */
struct resvd_st {
    int type;
    int value;
    char word[MAX_STRING_LENGTH];
};

typedef struct resvd_st resvd;

resvd reserved[NUM_RESERVED];

/* Current state quadruple */
typedef struct {
    int st, region, slice, point;
} cur_st;

cur_st current_state;

/* Saved slice array */
typedef struct {
    char name[MAX_STRING_LENGTH];
    int slice, region;
} slice_entry_st;

int slice_stk_ptr;
slice_entry_st slice_stack[STACK_SIZE];

/* Saved point array */
typedef struct {
    char name[MAX_STRING_LENGTH];
    int point, slice, region;
} point_entry_st;

int point_stk_ptr;
point_entry_st point_stack[STACK_SIZE];

/* Information about input string */
int token_type;
char token_str[128]; /* pointer to string if type is string */
int token_int; /* pointer to value if type is integer */
double token_dbl; /* pointer to value if type is double */
FILE *fp;
int mode; /* input mode: BATCH or INTERACTIVE */

extern int slice_chest();
extern int slice_seat();
extern int slice_waist();
extern int s_moveup();
extern int s_movedown();
extern int s_max();
extern int s_min();
extern int centerfront();
extern int centerback();
extern int centerleft();
extern int centerleft();

```

```
extern int leftmost();
extern int rightmost();
extern int foremost();
extern int hindmost();
extern int p_moveup();
extern int p_movedown();
extern int p_clockwise();
extern int p_counter_clockwise();
extern int max_X();
extern int min_X();
extern int max_Z();
extern int min_Z();
extern double circumference();
extern double direct_dist();
extern double surface_dist();
extern double lengthInches();

extern region_torso();
extern region_up_torso();
extern region_all();
```


manlps.h

Wed Mar 10 20:33:03 1993

1

include "common.h"

extern point3 *allPoints,*showPoints,rotOrigin;
extern int numPoints,speedUnit;
extern short incFlag,autoFlag;
extern point2 cop;

meas_menu.h

Sun Jun 27 14:30:50 1993

1

```
# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include <stdio.h>

# define meas_fg_red 0
# define meas_fg_blue 60000
# define meas_fg_green 60000

# define meas_bg_red 30000
# define meas_bg_blue 30000
# define meas_bg_green 30000

# define FN_WIDTH 25
# define MAX_MEASMTS 4
# define MAX_FILE_NAME 50
# define STR_LOC_X 20
# define STR_LOC_Y 15

extern Display *ncdisplay;
extern Window menu_window, ncwindow, meas_win, cur_fname_win;
extern GC cur_fname_gc;
extern XEvent ncevent;
extern long hbg, hfg;
extern int ncscreen;
extern Colormap nc_cmap;
```

measure.h

Wed Mar 10 20:33:05 1993

1

```
# include <math.h>

# include "common.h"

extern int      chest;
extern double   stdChest, stdInchChest;
extern point3   *allPoints;
extern int      allSlices[MAX_POINT_NUM];
```

```
# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include "pmeter.h"

# define BUTON 1
# define PMETER 2

# define MAX_BUTTONS 30
# define MAX_SEPARATORS 10
# define MAX_NAMES 10
# define MAX_STRING 80

# define BG_BUT 0
# define ROTATE_X_BUT 1
# define ROTATE_Y_BUT 2
# define ROTATE_Z_BUT 3
# define TRANS_X_BUT 4
# define TRANS_Y_BUT 5
# define TRANS_Z_BUT 6
# define AUTO_X_BUT 7
# define AUTO_Y_BUT 8
# define AUTO_Z_BUT 9
# define AUTO_RANDOM_BUT 10
# define SCALE_BUT 11
# define RESET_BUT 12
# define SLOWER_BUT 13
# define FASTER_BUT 14
# define DATA_FILES_BUT 15
# define AUTOSTOP_BUT 16
# define QUIT_BUT 17
# define MEASURE_BUT 18
# define SLICE_BUT 19

# define ROTATE_MET 1
# define TRANS_MET 2
# define SCALE_MET 3
# define AUTO_ROT_MET 4

struct menu_item
{
    int butt_no;
    int x,y,width,height;
    Region bounds;
    char title[80];
    void *action;
};

struct name
{
    int x,y;
    char name[MAX_STRING];
};

extern Display *ndisplay;
extern Window menu_window;
extern GC menu_gc;
extern unsigned long hbg,hfg;

extern struct pmeter pMeters[];
extern int pMeterCtr;
```

multi_meas.h

Wed Mar 10 20:33:06 1993

1

```
# include "common.h"
# include <X11/Xlib.h>
# include <X11/Xutil.h>
```

```
/*      max no. of segments in a sequence of clicks      */
# define MAX_SEGMENTS 20
```

```
extern Display *ncdisplay;
extern Window ncwindow;
```

```
extern int    numCurves;
extern int    pickSlc;
```

```

# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include <X11/keysymdef.h>
# include <X11/keysym.h>

# define MAX_PNETERS 10
# define PMETER_WIDTH 10
# define MAX_STR 40

# define PI 3.1415926535897932252626

/* structure definition for potentiometer */

struct pmeter {
    char pMName[MAX_STR];
    char unit;
    int pMNum;
    int x,y;
    int extent;
    int leftVal,rightVal;
    Region bounds,curvalReg;
    int curPosX,curPosY;
    int curVal;
};

extern Display *ncdisplay;
extern Window menu_window;
extern GC menu_gc,menu_xor_gc,menu_pm_gc;
extern unsigned long hbg,hfg;
extern XColor butCol1, butCol2;

```

```
# include "common.h"
# include <X11/Xlib.h>
# include <X11/Xutil.h>

# define MAX_CURVES 50

extern Display *ncdisplay;
extern Window ncwindow,meas_win;
extern GC nccg,meas_gc;

extern int numSegs,pickSlc;
extern point3 *showPoints;

extern double lengthInInches();
```

```
#include "disp.h"
#include "interpret.h"

#define L 0
#define R 1
#define EPS 0.01

#define R_ALL 0
#define R_ARM_L 1
#define R_ARM_R 2
#define R_LEG_L 3
#define R_LEG_R 4
#define R_TORSO 5
#define R_U_TORSO 6

#define ARM_CIRCUM 350.0
#define LEG_CIRCUM 550.0
#define TORSO_CIRCUM 600.0

typedef struct region_st {
    int nslices;
    int *slices;
} region;

region region_des;

point3 *slice_center;

extern int numPoints;
extern point3 *allPoints;
extern point3 *showPoints;
extern int numSlices;
extern int allSlices[MAX_POINT_NUM];
extern int rotOrigin;
extern double distance();
extern double realSliceLen();
extern double inchInLength();
extern double yz_dist();
extern cur_st current_state;
```



```
# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include <stdio.h>

# include "common.h"

# define slice_fg_RED 0
# define slice_fg_BLUE 60000
# define slice_fg_GREEN 60000

# define slice_bg_RED 30000
# define slice_bg_BLUE 30000
# define slice_bg_GREEN 30000

# define FN_WIDTH 25
# define MAX_VIEWS 3
# define MAX_FILE_NAME 50
# define STR_LOC_X 20
# define STR_LOC_Y 15

extern Display *ncdisplay;
extern Window menu_window, ncwindow, slice_win, cur_fname_win;
extern GC cur_fname_gc;
extern XEvent ncevent;
extern long hbg, hfg;
extern int ncscreen;
extern Colormap nc_cmap;
```

```
#include "disp.h"

extern int numPoints;
extern int allSlices[MAX_POINT_NUM];
extern int numSlices;
extern point3 *allPoints;

/*****
 * Routine defined
 * in measure.c
 *****/
extern double distance();

/*****
 * This structure is used in
 * dijkstra.C, it returns the
 * slice number which a point
 * belongs to.
 *****/
extern int inslice[MAX_POINT_NUM];

struct disp_pts_struct {
    int num;
    int *points;
};

typedef struct disp_pts_struct disp_pts;

struct collapse_slice_st {
    int num;
    int *slices;
    double circumference;
};

typedef struct collapse_slice_st collapse_slices;
```

```

# include <X11/Xlib.h>
# include <X11/Xutil.h>
# include "common.h"
# include "slice_view.h"
# include "region.h"

# define WIN_FG_RED 0
# define WIN_FG_BLUE 65535
# define WIN_FG_GREEN 65535

# define WIN_BG_RED 0
# define WIN_BG_BLUE 0
# define WIN_BG_GREEN 0

# define MENU_WIN_FG_RED 20000
# define MENU_WIN_FG_BLUE 20000
# define MENU_WIN_FG_GREEN 20000

# define MENU_WIN_BG_RED 40768
# define MENU_WIN_BG_BLUE 40768
# define MENU_WIN_BG_GREEN 40768

# define BUT_RED1 50000
# define BUT_BLUE1 51000
# define BUT_GREEN1 50000

# define BUT_RED2 30000
# define BUT_BLUE2 35000
# define BUT_GREEN2 30000

# define BUT_RED3 16784
# define BUT_BLUE3 65535
# define BUT_GREEN3 0

# define WIN_WIDTH 640
# define WIN_HEIGHT 640
# define MENU_WIN_WIDTH 310
# define MENU_WIN_HEIGHT 380
# define MEAS_WIN_WIDTH 310
# define MEAS_WIN_HEIGHT 220
# define CUR_FNAME_WIN_WIDTH 200
# define CUR_FNAME_WIN_HEIGHT 50

extern point3 *showPoints;
extern point3 *allPoints;
extern int allSlices[MAX_POINT_NUM];
extern int numPoints;
extern int chest, waist, seat;
extern int pickPnt[2], pickSlc;
extern int numSegs, numCurves;
extern double total_seq_len;
extern curve_dis cDistances[], radialDistances[];

extern int cur_axis, is_vert_slice, is_colp_slice;
extern disp_pts collaps_slice, vert_slice;

extern double sliceLen();
extern double pointLen();

extern XColor butCol1, butCol2, butCol3;
extern double lengthInInches();
extern double distance();

extern double distance();

extern point3 *slice_center;
extern region cregion_des;

extern int thefirst; /* remove it when center is not displayed. */

extern int slice_chest();
extern int slice_seat();
extern int slice_waist();
extern int s_moveup();
extern int s_movedown();
extern int s_max();
extern int s_min();
extern int centerfront();
extern int centerback();
extern int centerleft();
extern int centerleft();
extern int leftmost();
extern int rightmost();
extern int foremost();
extern int hindmost();
extern int p_moveup();
extern int p_movedown();
extern int p_clockwise();
extern int p_counter_clockwise();
extern int max_X();
extern int min_X();
extern int max_Z();
extern int min_Z();
extern double circumference();
extern double direct_dist();
extern double surface_dist();

```

transform.h

Fri Aug 13 15:27:11 1993

1

```
#include <math.h>
#include "disp.h"

#define STD_SP_SIZE 800
/** define PI 3.1415926535897932352626 */
#define TRANS_UNIT 20 /* translation unit */
/* # define ROTATE_UNIT PI/8.0 */ /* rotation unit */
#define SPEED_UNIT 100000

extern point2 cop;
extern double spSize;
extern short incFlag;
extern double COS30;
extern short incFlag;
```

```
# include "circum_meas.h"

void measSlice()
{
    XEvent sliceEvent;
    int done;

    numCurves = numSegs = 0;
    done = 0;
    while (!done)
    {
        XNextEvent( ncdisplay, &sliceEvent );

        switch (sliceEvent.type)
        {
            case ButtonPress :
                if (sliceEvent.xbutton.window != ncwindow)
                    break;

                if (sliceEvent.xbutton.button == 3) {
                    done = 1;
                    break;
                }

                if (sliceEvent.xbutton.button == 1)
                    showPickSlice(sliceEvent.xbutton.x, sliceEvent.x
button.y);

                break;

            default :
                break;
        }
    }
}
```

```

#include "dfiles.h"

char dfiles[ MAX_DFFILES ][ MAX_FILE_NAME ];
Window df_win, frame_wins[ MAX_DFFILES ];
GC df_gc, frame_gcs[ MAX_DFFILES ];
XColor fg_col, bg_col;
int frameCnt;
int cur_fname=5;

Initialize_DFMenu()
{
    int i;
    FILE *dfilesFp;
    XFontStruct *fontStr;

    dfilesFp = fopen("/u/pargas/3d/data-set/datafiles", "r");

    frameCnt = 0;
    while ( ! fgets(dfiles[frameCnt], MAX_FILE_NAME, dfilesFp) != NULL ) {
        i=0;
        while ( dfiles[frameCnt][i++] != '\0' );
        dfiles[frameCnt][i-2] = '\0';
        frameCnt++;
    }
    fclose(dfilesFp);

    create_dfile_windows(&df_win, &df_gc, frame_wins, frame_gcs, frameCnt);

    fontStr = XLoadQueryFont( ncdisplay, "-adobe-courier-bold-r-normal--0-0-75-75-m-0-iso8859-1" );
    if (fontStr != 0)
        for (i=0; i<frameCnt; i++)
            XSetFont( ncdisplay, frame_gcs[i], fontStr->fid);
}

refresh_DFMenu()
{
    int i;

    for (i=0; i<frameCnt; i++)
        XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_LOC_Y, dfiles[i],
            strlen(dfiles[i]));
        XDrawString(ncdisplay, frame_wins[frameCnt], frame_gcs[0], STR_LOC_X, STR_LOC_Y, "Done",
            strlen("Done"));
}

map_DFMenu(x,y)
int x,y;
{
    int i;
    XEvent df_event;
    XWindowAttributes winAttrs;

    XMapRaised(ncdisplay, df_win);
    for (i=0; i<frameCnt+1; i++)
        XMapRaised(ncdisplay, frame_wins[i]);

    XGetWindowAttributes( ncdisplay, frame_wins[frameCnt], &winAttrs );

    if (winAttrs.map_state != 2)
        XWindowEvent( ncdisplay, frame_wins[frameCnt], ExposureMask, &df_event );

    redraw_DFMenu();
}

redraw_DFMenu()
{
    int i;

    for (i=0; i<frameCnt; i++)
        XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_LOC_Y, dfiles[i],
            strlen(dfiles[i]));
        XDrawString(ncdisplay, frame_wins[frameCnt], frame_gcs[0], STR_LOC_X, STR_LOC_Y, "Done",
            strlen("Done"));
        XFlush( ncdisplay );
}

dataFiles()
{
    int i;
    char filename[ MAX_FILE_NAME ];

    switch (nccurrent.type)
    {
        case Expose :
            for (i=0; i<frameCnt+1; i++)
                if (nccurrent.xexpose.window == frame_wins[i]) break;
            if ( i < frameCnt+1 ) {
                for (i=0; i<frameCnt; i++)
                    XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_LOC_Y, dfiles[i],
                        strlen(dfiles[i]));
                    XDrawString(ncdisplay, frame_wins[frameCnt], frame_gcs[0], STR_LOC_X, STR_LOC_Y, "Done",
                        strlen("Done"));
            }
            break;

        case MapNotify :
            if (nccurrent.xmap.window == df_win)
                for (i=0; i<frameCnt; i++) {
                    XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_LOC_Y, dfiles[i],
                        strlen(dfiles[i]));
                }
                XDrawString(ncdisplay, frame_wins[frameCnt], frame_gcs[0], STR_LOC_X, STR_LOC_Y, "Done",
                    strlen("Done"));
            break;

        case ButtonPress :
            for (i=0; i<frameCnt; i++)
                if (nccurrent.xbutton.subwindow == frame_wins[i]) break;
            if ( i == frameCnt ) {
                XUnmapWindow( ncdisplay, df_win );
                XFlush( ncdisplay );
            }
            else {
                XSetForeground( ncdisplay, frame_gcs[i], bg_col.pixel );
                XSetWindowBackground( ncdisplay, frame_wins[i], fg_col.pixel );
                XClearWindow( ncdisplay, frame_wins[i] );
            }
}

```

```

OC_Y, dfiles[i], strlen(dfiles[i]));
XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_L
_X, dfiles[i], strlen(dfiles[i]));
XFlush(ncdisplay);

cur_fname = i;
strcpy(filename, dfiles[i]);
manage_datafile(filename);

XSetForeground(ncdisplay, frame_gcs[i], fg_col.pixel);
XSetWindowBackground(ncdisplay, frame_wins[i], bg_col.pixel);
XClearWindow(ncdisplay, frame_wins[i]);
XDrawString(ncdisplay, frame_wins[i], frame_gcs[i], STR_LOC_X, STR_L
_Y, dfiles[i], strlen(dfiles[i]));
XFlush(ncdisplay);
}

break;
default :
break;
)

)

create_dfile_windows(win_gc, frame_wins, frame_gcs, num_files)
Window *win, *frame_wins;
GC *gc, *frame_gcs;
int num_files;
{
    XSetWindowAttributes dfile_win_attrs;
    unsigned long dfile_win_mask;
    XSizeHints dfile_hint, frame_hints[MAX_DFILES];
    int i, depth;

    depth = XDisplayPlanes(ncdisplay, nc_screen);
    bg_col.red = DF_BG_RED;
    bg_col.blue = DF_BG_BLUE;
    bg_col.green = DF_BG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &bg_col);

    fg_col.red = DF_FG_RED;
    fg_col.blue = DF_FG_BLUE;
    fg_col.green = DF_FG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &fg_col);

    dfile_win_attrs.border_pixel = fg_col.pixel;
    dfile_win_attrs.background_pixel = bg_col.pixel;
    dfile_win_attrs.override_redirect = True;

    dfile_win_mask = ( CWBorderPixel | CWOverrideRedirect );

    /* set the window parameters for dfile window */
    dfile_hint.x = 800; dfile_hint.y = 450;
    dfile_hint.width = 120; dfile_hint.height = (num_files+1) * FN_WIDTH;
    dfile_hint.flags = USPosition | PSize;

    /* Create the window with the above given specifications */
    *win = XCreateSimpleWindow(ncdisplay, DefaultRootWindow(ncdisplay), dfile_h
int.x, dfile_hint.y,

```

```

#include "dijkstra.h"

int get_close();
int valid();
int point_mgn();
int h_insert();
int h_up();
SPAN_TREE_ST h_deletemin();

/* Fast access of slice number each point belongs to */
int inslice[MAX_POINT_NUM];

comp_path(source, sink, curve)
int source, sink;
curve_dis *curve;
{
    int span_count;
    int not_done;
    int i, j;

    if( source<0 || sink<0 || source>=numPoints || sink>=numPoints )
        return;
    init_tree(source);
    span_count = -1;
    not_done = 1;
    while( not_done ){
        span_count++;
        /*span(span_count); */
        get_close(span_count);
        span_tree[span_count] = h_deletemin();
    }
    /*printf("%d %d %f %\n", span_tree[span_count].num, span_tree[span_count].wgt, span_tree[sp
    an_count].pred); */
    marked[span_tree[span_count].num] = span_count;
    highlight_point(span_tree[span_count].num);
    if( span_tree[span_count].num==sink ) not_done=0;
}

/*printf("\n distance is %f\n", span_tree[span_count].wgt); */
/* processing result */
curve->src = source;
curve->sink = sink;
curve->ds = span_tree[span_count].wgt;
i = span_tree[span_count].pred;
curve->path[0] = sink;
j = 1;
while( i!=0 ){
    curve->path[j] = span_tree[i].num;
    i = span_tree[i].pred;
    j++;
}
curve->path[j] = source;
curve->length = j;
}

init_tree(source)
int source;
{
    int i, j, k;
    int p;

    part_last = 0; /* init partial tree for heap sort */
    part_tree[0] = 0;
    temp_last = 0;

    for( i=0; i<numPoints; i++ ){ /* init spanning tree */
        marked[i] = -1;
        span_tree[i].num = -1;
        span_tree[i].pred = -1;
        span_tree[i].count = 0;
        mark_temp[i] = -1;
        span_temp[i].num = -1;
        span_temp[i].pred = -1;
    }

    span_temp[temp_last].num = source;
    span_temp[temp_last].wgt = 0.0;
    span_temp[temp_last].pred = -1;
    span_temp[temp_last].count = 0;

    p = 0;
    j = k = 0;
    for( i=0; i<numSlices; i++ ){
        for( k=allSlices[i+1]-j; k>0; k-- )
            inslices[p++] = i;
        j = allSlices[i+1];
    }

    if( p != numPoints ) printf("Dijkstra init error\n");

    /******
    * span the entryTH node of the temporary tree.
    * *****/
    span_part(entry, num, wgt, pred)
    int entry, num, pred;
    double wgt;
    {
        span_temp[entry].num = num;
        span_temp[entry].wgt = wgt;
        span_temp[entry].pred = pred;
        /*marked[num] = count; */

        update_heap(i, wgt, span_count)
        int i, span_count;
        double wgt;
        {
            if( mark_temp[i] == -1 ){
                span_part(++temp_last, i, wgt, span_count);
                mark_temp[i] = h_insert(i, temp_last);
                if( span_temp[part_tree[mark_temp[i]]].num != i ){
                    printf("error not matching 0"); /*getchar(); */
                }
            }
            else
                if( wgt < span_temp[part_tree[mark_temp[i]]].wgt ){
                    if( span_temp[part_tree[mark_temp[i]]].num != i ){
                        printf("error not matching 1"); /*getchar(); */
                    }
                    span_temp[part_tree[mark_temp[i]]].wgt = wgt;
                    span_temp[part_tree[mark_temp[i]]].pred = span_count;
                    mark_temp[i] = h_up(mark_temp[i]);
                    if( span_temp[part_tree[mark_temp[i]]].num != i ){
                        printf("error not matching 2"); /*getchar(); */
                    }
                }
        }

    /******
    Get close returns the closest candidate to
    father. Distance from source is specified in weight.
    */

```



```

*****
int get_close(span_count)
int span_count;
{
    int i, j;
    double d, f_wgt, wght;
    int father;

    father = span_temp[part_tree[0]].num;
    f_wgt = span_temp[part_tree[0]].wgt;
    /* if( span_tree[marked[father]].count==(-1) ) return(0); */
    i = point_mgn(father, (-1)); /* left */
    if( valid(i, father, &d) ){
        wght = f_wgt + d;
        update_heap(i, wght, span_count);
    }
    i = point_mgn(j, (-1)); /* lower center */
    if( valid(i, father, &d) ){
        wght = f_wgt + d;
        update_heap(i, wght, span_count);
    }
    i = j; /* lower right */
    if( valid(i, father, &d) ){
        wght = f_wgt + d;
        update_heap(i, wght, span_count);
    }
    j = adjacent[0][father];
    if ( j > (-1) ){
        i = point_mgn(j, (-3));
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = point_mgn(j, (-5));
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = point_mgn(j, (-1)); /* upper left */
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = j; /* upper center */
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = point_mgn(j, 1); /* upper right */
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = point_mgn(j, 5);
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        i = point_mgn(j, 3);
        if( valid(i, father, &d) ){
            wght = f_wgt + d;
            update_heap(i, wght, span_count);
        }
        j = adjacent[1][father];
        if ( j > (-1) ){
            i = point_mgn(j, (-3));
            if( valid(i, father, &d) ){
                wght = f_wgt + d;
            }
        }
    }
}

update_heap(i, wght, span_count);

}

int point_mgn(j, (-5));
if( valid(i, father, &d) ){
    wght = f_wgt + d;
    update_heap(i, wght, span_count);
}

int point_mgn(j, 5);
if( valid(i, father, &d) ){
    wght = f_wgt + d;
    update_heap(i, wght, span_count);
}

int point_mgn(j, 3);
if( valid(i, father, &d) ){
    wght = f_wgt + d;
    update_heap(i, wght, span_count);
}

return(1);
}

/* The routine first check if p1 is a valid entry
 * and not yet expanded to span_tree[]
 * in allPoints[]. If so, returns the distance
 * between p1 and p2 from ds.
 */
int valid(p1, p2, ds)
int p1, p2;
double *ds;
{
    if( marked[p1] != -1 ) return(0);
    *ds = 1000000.0;
    if( p1 < 0 || p1 >= numPoints || marked[p1] > (-1) ) return(0);
    else{
        *ds = distance(allPoints[p1], allPoints[p2]);
        return(1);
    }
}

/* This routine returns a point that is a given
 * margin index from the current point and is in
 * the same slice.
 */
int point_mgn(p, margin)
int p, margin;
{

```

```

int i, j;
int low, high;

j = p-margin;
if( inslice[j]==inslice[p] ) return(j);
i = inslice[p];
low = allslices[i];
high = allslices[i+1];
j = ((p-low) + margin) % (high-low) + low;
return(j);
}

/*****
 * Heap sort insert
 * Parameters:
 * point-- the entry in allPoints[]
 * node -- entry in span_temp[]
 *****/
int h_insert(point, node)
int node, point;
{
    /* if( marked[node] != (-1) ) return(-1); */
    if( span_temp[node].num != point ) {
        printf("Span not matching in h_insert\n");
        getchar();
    }
    mark_temp[point] = ++part_last;
    part_tree[part_last] = node;
    return(h_up(part_last));
}

/*****
 * Heap sort: float a changed-vlaue
 * node up in the tree
 * (new value is smaller)
 *****/
int h_up(i)
int i;
{
    int temp;

    while( i>0 && span_temp[part_tree[i]].wgt<span_temp[part_tree[(i-1)/2]].wgt ) {
        swap_int( &mark_temp[span_temp[part_tree[i]].num],
                  &mark_temp[span_temp[part_tree[(i-1)/2]].num] );
        swap_int( &part_tree[i], &part_tree[(i-1)/2] );
        i = (i-1)/2;
    }
    return(i);
}

/*****
 * heap sort delete minimum
 *****/
SPAN_TREE_ST h_deletemin()
{
    int i, j;
    SPAN_TREE_ST temp_node;

    temp_node = span_temp[part_tree[0]];
    part_tree[0] = part_tree[part_last-1];
    mark_temp[span_temp[part_tree[0]].num] = 0;
    i = 0;
    while( i<=((part_last-1)/2) ) {

```

```

        if( (span_temp[part_tree[2*i+1]].wgt<span_temp[part_tree[2*i+2]].wgt)
            || 2*i+1==part_last )
            j = 2*i+1;
        else j = 2*i+2;
        if( span_temp[part_tree[i]].wgt > span_temp[part_tree[j]].wgt ) {
            swap_int( &mark_temp[span_temp[part_tree[i]].num],
                      &mark_temp[span_temp[part_tree[j]].num] );
            swap_int( &part_tree[i], &part_tree[j] );
            i = j;
        }
        else return(temp_node);
    }
    return(temp_node);
}

swap_int(a,b)
int *a, *b;
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;
}

```

```

#include "disp.h"
#include "menu.h"

double COS30;
point3 rotOrigin;
point3 *showPoints;

point2 cop;
double spsize;
extern int numPoints;

short incFlag=0; /* "*" or "--" flag */
short autoFlag=NO_AUTO_ROTATE;
short speedUnit;
extern double stdChest, stdInchChest; /* Just change! jan 24 */

#ifdef SUN_VERSION
XPoint *scr_coords;
XEvent ncevent;
int menuNum, menuCl;

/* array of screen (2-d) points */
#endif

main(argc,argv)
int argc;
char **argv;
{
    #ifdef SUN_VERSION
    KeySym nckey;
    XKeyEvent kevent;
    XMappingEvent mapevent;

    int i,j,choice;
    char text[80];
    int x,y;
    /* int menuNum,menuCl; */
    int long event_mask;

    int expose_cnt=0;

    /* Initialize the dimensions and various properties of the window */
    Initialize_X();
    Initialize_windows();

    /* Initialize the transformation matrices and calculate the
    /* the viewing transformation matrix

    if (argc == 1) Initialize("data.dat");
    else Initialize(argv[1]);

    Initialize_Menu();
    Initialize_DPMenu();
    Initialize_measMenu();
    Initialize_sliceMenu();

    event_mask = ExposureMask | ButtonPressMask | EnterWindowMask |
    KeymapStateMask | ButtonReleaseMask;

    /* The permanent event checking iteration */
    while (1) {
        if ((XCheckMaskEvent( ncdisplay, event_mask, &ncevent ) != 0))
        {
            switch (ncevent.type) {
                case Expose :
                    processExposeEvent(&ncevent);
                    break;

                case MappingNotify :
                    mapevent = ncevent.xmapping;
                    XRefreshKeyboardMapping (&mapevent);
                    break;

                /* When a mouse button is pressed the appropriate transformation */
                /* is applied to the existing figure and redisplayed */

                case ButtonPress :
                    if (ncevent.xbutton.window == menu_window) {
                        processMenuPress(&ncevent);
                    }
                    /*
                    check_menu_item(ncevent.xbutton.x,ncevent.xbutton
                        .y,&menuNum,&menuCl);

                    switch (menuCl) {
                        case BUTTON :
                            if (menuNum != 0)
                                inverse(menuNum,0);
                                processButtons( ncevent, menuNum
                                    break;

                                case PMETER :
                                    XSetInputFocus(ncdisplay, menu_wi
                                        ndow, RevertToParent, ncevent.xcrossing.time);
                                    processMeters( ncevent.xbutton.x,
                                        ncevent.xbutton.y, menuNum );
                                    break;

                                /*
                                XFlush (ncdisplay);
                                */
                                }
                                else
                                    if (ncevent.xbutton.window == df_win)
                                        dataFiles();
                                else
                                    if (ncevent.xbutton.window == measMenu_win)
                                        measMenu();
                                else
                                    if (ncevent.xbutton.window == sliceMenu_win)
                                        sliceMenu();
                                /*
                                if ( ncevent.xbutton.window==ncwindow && ncevent.xbutton-
                                    button==3)
                                        showPicksSlice(ncevent.xbutton.x, ncevent.xbutton.
                                            y);
                                else
                                    if ( ncevent.xbutton.window==ncwindow && ncevent.xbutton-
                                        button==1 )
                                            showPickPoint(ncevent.xbutton.x, ncevent.xbutton.
                                                Y, 1);
                                else
                                    if ( ncevent.xbutton.window==ncwindow && ncevent.xbutton.

```

```

button==2 )
.y, 2);
*/
showPickPoint(ncevent.xbutton.x, ncevent.xbutton
break;
case ButtonRelease :
if ((menuNum != 0) && (menuCl == BUTTON))
inverse(menuNum,1);
break;
case EnterNotify :
XSetInputFocus(ncdisplay, menu_window, RevertToParent, n
cevent.xcrossing.time);
break;
)
)
else
if (autoFlag != NO_AUTO_ROTATE)
{
update_scr(autoFlag);
delay();
}
} else
/*
For the PC version
code for the initialization and main event processing loop to be inserted.
*/
# endif
)
/* This routine gives the user a choice to manipulate the already displayed 3D
/* figure. This is infinite iteration unless you choose 'QUIT' to exit from the
/* system
processButtons( event, choice )
XEvent event;
int choice;
{
/* Add the appropriate transformation to the matrix depending on the choice */
switch( choice )
{
case BG_BUTTON :
break;
case ROTATE_X_BUTTON :
rotX(rotateUnit);
break;
case ROTATE_Y_BUTTON :
rotY(rotateUnit);
break;
case ROTATE_Z_BUTTON :
rotZ(rotateUnit);
break;
case TRANS_X_BUTTON :
transX();
break;
case TRANS_Y_BUTTON :
transY();
break;
case TRANS_Z_BUTTON :
transZ();
break;
case SCALE_BUTTON :
scaler(scaleUnit);
break;
case RESET_BUTTON :
reset();
break;
case AUTO_X_BUTTON :
autoFlag = X_AXIS;
break;
case AUTO_Y_BUTTON :
autoFlag = Y_AXIS;
break;
case AUTO_Z_BUTTON :
autoFlag = Z_AXIS;
break;
case AUTO_RANDOM_BUTTON :
autoFlag = RANDOM;
break;
case SLOWER_BUTTON :
chSpeed(0);
break;
case FASTER_BUTTON :
chSpeed(1);
break;
case AUTOSTOP_BUTTON :
autoFlag = NO_AUTO_ROTATE;
break;
case QUIT_BUTTON :
exit(0);
break;
case DATA_FILES_BUTTON :
mapDFMenu(800,450);
break;
case MEASURE_BUTTON :
mapmeasMenu(800,450);
break;
case SLICE_BUTTON :
mapsliceMenu(800,450);
break;

```

```

    }

processMeters( x,y,choice )
{
    int x,y;
    int choice;

    switch ( choice )
    {
        case ROTATE_MET :
            changeAngle( x,y,&rotateUnit,choice-1 );
            break;

        case TRANS_MET :
            changeTrans( x,y,&transUnit,choice-1 );
            break;

        case SCALE_MET :
            changeScale( x,y,&scaleUnit,choice-1 );
            break;

        case AUTO_ROT_MET :
            changeSpeed( x,y,&speedUnit,choice-1 );
            break;
    }
}

update_scr(autoFlag)
{
    static int cnt=0;
    int axis;

    if ( autoFlag == NO_AUTO_ROTATE )
        return;

    if ( autoFlag == RANDOM )
        axis = rand()%3;
    else
        axis = autoFlag;

    switch (axis)
    {
        case X_AXIS : rotX(rotateUnit);
            break;
        case Y_AXIS : rotY(rotateUnit);
            break;
        case Z_AXIS : rotZ(rotateUnit);
            break;
    }

    delay()

    usleep((100 - speedUnit)*20000);

processButtonClick(event)
{
    Event *event;

    int menuNum, menuCl;

    if (event->xbutton.window == menu_window) {
        check_menu_item(event->xbutton.x,event->xbutton.y,&menuNum,&menuCl);
        switch (menuCl) {
            case BUTTON :
                if (menuNum != 0)
                    inverse(menuNum,0);
                processButtons( event, menuNum );
                break;

            case PMETER :
                XSetInputFocus(ncdisplay, menu_window, RevertToParent, event->xcrossing.time);
                processMeters( event->xbutton.x, event->xbutton.y, menuNum );
                break;

            case XFLUSH : ncdisplay );
        }

        XNextEvent( ncdisplay, event );
        while (event->type != ButtonRelease )
            XNextEvent( ncdisplay, event );
        if ( (menuNum != 0) && (menuCl == BUTTON) )
            inverse(menuNum,1);
    }
}

processExposeEvent(event)
{
    XEvent *event;

    if (event->xexpose.window == ncwindow)
        display();
    else if (event->xexpose.window == menu_window)
        redrawMenu();
    else if (event->xexpose.window == meas_win)
        dispMeasnts();
    else if (event->xexpose.window == fname_wins[fnameCnt])
        redrawDFMenu();
    else if (event->xexpose.window == meas_wins[measCnt])
        redrawmeasMenu();
    else if (event->xexpose.window == slice_wins[sliceCnt])
        redrawsliceMenu();
    else if (event->xexpose.window == cur_fname_win)
        dispFName();
}

processMenuPress(event)
{
    XEvent *event;

    check_menu_item(event->xbutton.x,event->xbutton.y,&menuNum,&menuCl);
    switch (menuCl) {
        case BUTTON :
            if (menuNum != 0)
                inverse(menuNum,0);
            processButtons( event, menuNum );
            break;

        case PMETER :
    }
}

```

ilsp.c

Wed Aug 25 09:24:24 1993

4

```
crossing.time);
    }
    XFlush (ncdisplay);
    XSetInputFocus(ncdisplay, menu_window, RevertToParent, event->xc
    processMeters( event->xbutton.x, event->xbutton.y, menuNum );
    break;
```

```

#include "disp_slices.h"

int
disp_pts
int
{
    int i, point;
    XEvent event;
    int done;
    int slice_no;

    is_vert_slice = 0;
    is_colp_slice = 0;

    display();

    slice_no = inslice[point];
    done = 0;
    while (!done) {
        XNextEvent( ncdisplay, &event );
        switch (event.type) {

        case Expose :
            processExposeEvent(&event);
            break;

        case ButtonPress :
            XSetInputFocus( ncdisplay, ncwindow, RevertToParent, event.xcr
ossing.time);

            is_vert_slice = 1;
            if (event.xbutton.window == ncwindow) {
                if (event.xbutton.button == 3) {
                    done = 1;
                    break;
                }

                point = pickPoint(event.xbutton.x, event.xbutton.y);
                if (point == -1)
                    break;

                switch (axis) {
                    case Y_AXIS :
                        cur_axis = Y_AXIS;
                        slice_view_vertical(point, &vert_slice, Y_AXIS);
                        break;

                    case Z_AXIS :
                        cur_axis = Z_AXIS;
                        slice_view_vertical(point, &vert_slice, Z_AXIS);
                        break;

                    default :
                        break;
                }

                display();
            }
            else {
                done = 1;
            }
        }
    }
}

/* is_vert_slice = 0; */
break;
case KeyPress :
    processKeyEvent( &event, &point, &slice_no, axis );
    break;

default :
    break;
}

}

processKeyEvent( event, point, slice_no, axis)
XEvent *event;
int *point, *slice_no;
int axis;
{
    int count, status;
    char text[2];
    KeySym key;

    count = XLookupString(&(event->xkey), text, 2, &key, &status);
    switch (key) {
        case XK_Up :
            if (axis != Z_AXIS) return;
            (*point) = (*point) + 2;
            if (inslice[*point] != *slice_no)
                *point = allSlices[*slice_no];
            break;

        case XK_Left :
            if (axis != Y_AXIS) return;
            (*point)--;
            if (inslice[*point] != *slice_no)
                *point = allSlices[*slice_no+1] - 1;
            break;

        case XK_Right :
            if (axis != Y_AXIS) return;
            (*point)++;
            if (inslice[*point] != *slice_no)
                *point = allSlices[*slice_no];
            break;

        case XK_Down :
            if (axis != Z_AXIS) return;
            (*point) = (*point) - 2;
            if (inslice[*point] != *slice_no)
                *point = allSlices[*slice_no+1] - 1;
            break;

        default :
            return;
    }

    switch (axis) {
        case Y_AXIS :
            slice_view_vertical(*point, &vert_slice, Y_AXIS);
    }
}

```

```

    break;
}

case Z_AXIS :
    slice_view_vertical(*point, &vert_slice, Z_AXIS);
    break;

default :
    break;

}

display();

}

display_collapsed_slices()
{
    XEvent event;
    int i, point, done;
    static int coll_count;
    collapse_slices collSlcs;
    int collapsed_slices[50];

    is_vert_slice = 0;
    is_colp_slice = 0;

    display();

    done = 0;
    coll_count = 0;
    while (!done)
    {
        XNextEvent(&ncdisplay, &event);
        switch (event.type)
        {
            case ButtonPress :
                if (event.xbutton.window != ncwindow)
                    break;

                if (event.xbutton.button == 3) {
                    done = 1;
                    collSlcs.num = coll_count;
                    collSlcs.slices = (int *)
                        malloc( coll_count * sizeof(int) );
                    for (i=0; i<coll_count; i++)
                        collSlcs.slices[i] = collapsed_slices[i];

                    collapse(&collSlcs, &collaps_slice);
                    is_colp_slice = 1;
                    showCollpSlices(&collaps_slice);
                }

                if (event.xbutton.button == 1) {
                    collapsed_slices[coll_count] =
                        pickSlice(event.xbutton.x, event.xbutton.y);
                    if (collapsed_slices[coll_count] == -1)
                        break;
                    highlight(collapsed_slices[coll_count++]);
                }

            default :
                break;
        }
    }
}

```



```

/*****
This file contains routines dealing with file I/O.
*****/

#include "filein.h"

point3 *allPoints; /* allPoints stores all the positions of the points. */
int numPoints=0;
point3 *temp;

int allSlices[MAX_POINT_NUM];
int numSlices; /* Number of points. */

char buffer[BUFFER_SIZE]; /* Input buffer */
int counter, last; /* Use to record position in the input buffer. */

int chest, waist, seat;

double cvrtDouble();

/*****
Main input processing routine.
*****/
readall(fname)
char *fname;
{
    static first=1;
    int n,i;
    char c;
    FILE *fid;
    int number;
    double aNum, expCount;
    double symFlag=1.0;
    double *tempPtr;

    fid = fopen(fname,"r");
    if (fid == NULL) return(-1);

    counter = BUFFER_SIZE;
    last = BUFFER_SIZE;

    if (first) {
        allPoints = (point3 *) malloc(MAX_POINT_NUM * (sizeof *allPoints));
        first=0;
    }

    number = 0;
    tempPtr = (double *) allPoints;

    chest = 14;
    waist = 38;
    seat = 64;

    n = readIn(fid, &c);

    numSlices = 0;
    allSlices[numSlices] = 0;
    while(c != EOF && n>0) {
        if (c == 'S') {
            numSlices++;
            allSlices[numSlices] = number/3;
        }

        if (counter == last) {
            if (counter == last) {
                last = fread(buffer, sizeof(char), BUFFER_SIZE, fid);
                if (last == 0) return(0);
                counter = 0;
            }
            *c = buffer[counter];
            counter++;
        }

        if (c == '0') {
            aNum = cvrtDouble(c);
            expCount = 1.0;
            n = readIn(fid, &c);
            while( (c>='0') && (c<='9') ) {
                aNum = aNum*10.0 + cvrtDouble(c);
                n = readIn(fid, &c);
            };
            if (c == '.' ) {
                expCount = 0.1;
                n = readIn(fid, &c);
                while( (c>='0') && (c<='9') ) {
                    aNum *= expCount*cvrtDouble(c);
                    expCount *= 0.1;
                    n = readIn(fid, &c);
                }
            };
            tempPtr[number] = aNum*symFlag;
            number++;
        } else if ( c == '-' ) {
            symFlag = (-1.0);
            n = readIn(fid, &c);
        } else {
            symFlag = 1.0;
            n = readIn(fid, &c);
        }
        /* while c != EOF */
        numPoints = number/3;
        allSlices[numSlices] = numPoints;
    }

    /*****
    This routine converts a character digit into
    a double value.
    *****/
    double cvrtDouble(c)
    char c;
    {
        double d;

        d = (double) ( (int) c - (int) '0' );
        return(d);
    }

    /*****
    This routine returns a character from the
    data file. It returns 0 if EOF, otherwise
    returns 1. This is the routine that
    actually performs file input. It reads
    in a cluster each time for the reason
    of efficiency.
    *****/
    int readIn(fid, c)
    FILE *fid;
    char *c;
    {
        if ( (counter == last) && (last < BUFFER_SIZE) ) return(0);
        if ( counter == last ) {
            last = fread(buffer, sizeof(char), BUFFER_SIZE, fid);
            if ( last == 0 ) return(0);
            counter = 0;
        };
        *c = buffer[counter];
        counter++;
    }
}

```

filein.c

Thu Feb 24 13:01:31 1994

2

return(1);

)

```

#include "disp.h"
#define SLICE_RANGE 5
extern point3 *showPoints;
extern int numPoints;
extern int allSlices[MAX_POINT_NUM];
extern int numSlices;
extern double distance();

int adjacent[2][MAX_POINT_NUM]; /* adjacency matrix for each point */
/* entry 0, is to upper slice,
   entry 1, is to lower slice,
   also assume the points left and right to the up/low
   adjacent point is also adjacent to the current point.
   default adjacent includes the left and right points
   of same slice.
   -1 indicates a non-adjacency */

int getClosest();
init_graph()
{
    int i, j;
    for( i=0; i<2; i++)
        for( j=0; j<MAX_POINT_NUM; j++)
            adjacent[i][j] = -1;
    printf("finish init\n");
    graph();
}

/*****
 * This routine forms the adjacency graph
 * for each point. Currently, we compute
 * every point. A faster algorithm is
 * possible.
 *****/
graph()
{
    int i, j;
    int f_ran, b_ran;
    int qty;

    for( i=0; i<numSlices; i++) {
        if( i<SLICE_RANGE ) b_ran = i;
        else b_ran = SLICE_RANGE;
        if( (numSlices-i)<SLICE_RANGE ) f_ran = numSlices-i-1;
        else f_ran = SLICE_RANGE;
        qty = allSlices[i+1] - allSlices[i];
        for( j=0; j<qty; j++) find_adj(i, j, f_ran, b_ran);
    }
    printf("computed all %d points in slice %d\n", qty, i);
}

/*****
 * This routine convert the position of
 * the point we are currently interested
 * and the range we consider from slice
 * entry position to the 3D point entry
 * position.
 *****/

```

```

 * sliceNo, f_ran and b_ran are referred
 * to allSlices[]. pointNo is the
 * offset in sliceNo of the current point.
 *****/
find_adj(sliceNo, pointNo, f_ran, b_ran)
int sliceNo, pointNo; /* point entry from sliceNo and offset. */
int f_ran, b_ran; /* forward and backward range */
{
    int j;
    int start, end;
    int pointEntry;

    /*printf("\tnow slice=%d point=%d\n", sliceNo, pointNo);*/
    pointEntry = allSlices[sliceNo]+pointNo;
    if( b_ran != 0 ) { /* adjacent to upper slice */
        start = allSlices[sliceNo-b_ran];
        end = allSlices[sliceNo] - 1;
        j = getClosest(pointEntry, start, end);
        adjacent[0][pointEntry] = j;
    }
    if( f_ran != 0 ) { /* adjacent to lower slice */
        start = allSlices[sliceNo+1];
        end = allSlices[sliceNo+f_ran+1] - 1;
        j = getClosest(pointEntry, start, end);
        adjacent[1][pointEntry] = j;
    }
}

/*****
 * This routine returns the point whose 3D
 * distance is closest to the point indicated
 * by pointEntry, among all the points ranged
 * from start to end.
 * All entries are referred to allPoints.
 *****/
int getClosest(pointEntry, start, end)
int pointEntry, start, end;
{
    double dt1, dt2;
    int i, entry;

    /* Need to evaluate if the commented code is faster. */
    /*i = end;
    while( i>=start ) {
        if( (allPoints[i].y==allPoints[pointEntry].y) &&
            (abs(allPoints[i].z-allPoints[pointEntry].z)<20.0) )
            return(i);
        i--;
    }*/
    /* get an initial value */
    entry = end;
    dt1 = distance(allPoints[pointEntry], allPoints[end]);
    /*printf("get close: point=%d, start=%d, end=%d\n", pointEntry, start, end);*/
    for( i=start; i<end; i++) {
        dt2 = distance(allPoints[pointEntry], allPoints[i]);
        if( dt2<dt1 ) {
            entry = i;
            dt1 = dt2;
        }
    }
    return(entry);
}

```



```

#include "init.h"

int thefirst; /* will be removed if center if not displayed. */

/*****
Initialization of data.
*****/
Initialize(fname)
char *fname;
{
    static first=1;
    int i;
    double x,y,z,initAngle;

    if (readAll(fname) == -1) {
        fprintf(stderr,"File %s not found.\n",fname);
        exit(0);
    }
    if ( readGraph(fname) == -1) {
        fprintf(stderr,"File %s graph file not found.\n",fname);
        exit(0);
    }

    rotateUnit = 30.0*PI/180.0;
    scaleUnit = 25.0;
    speedUnit = 100;

    COS30 = (double) cos(PI/6.0);
    cop.x = 0.2*STD_SP_SIZE;
    cop.y = 0.5*STD_SP_SIZE;
    spSize = STD_SP_SIZE;
    incFlag = DECREASE;

    if (first) {
        scr_coords = (Xpoint *) malloc(MAX_POINT_NUM * (sizeof *scr_coords));
        showPoints = (point3 *) malloc(MAX_POINT_NUM * (sizeof *showPoints));
        set_meas_standard();
        first = 0;
    }

    calculate_measmts();

    x = 0.0;
    y = 0.0;
    z = 0.0;
    for( i=0; i<numPoints; i++) {
        x += allPoints[i].x;
        y += allPoints[i].y;
        z += allPoints[i].z;
        showPoints[i] = allPoints[i];
    };

    rotOrigin.x = x / numPoints;
    rotOrigin.y = y / numPoints;
    rotOrigin.z = z / numPoints;

    initAngle = 90.0*PI/180.0;
    for ( i=0; i<numPoints; i++)
        rotate(Z_AXIS, initAngle, &showPoints[i], &rotOrigin );
    scale(-300.0);
    transUnit = -130;
    translate(X_AXIS);
}

transUnit = -100;
translate(Y_AXIS);
transUnit = 25;
centers(showPoints); thefirst=1; /*this line will be removed when
not displaying the centers. Variable thefirst will also be removed.*/
}

/*****
Reset all the vieing parameters. Object resumes
its initial view.
*****/
void
reset()
{
    int i;
    double initAngle;

    for( i=0; i<numPoints; i++)
        showPoints[i] = allPoints[i];

    COS30 = (double) cos(PI/6.0);
    cop.x = 0.2*STD_SP_SIZE;
    cop.y = 0.5*STD_SP_SIZE;
    spSize = STD_SP_SIZE;
    incFlag = DECREASE;

    autoFlag = NO_AUTO_ROTATE;
    speedUnit = 100;

    initAngle = 90.0*PI/180.0;
    for ( i=0; i<numPoints; i++)
        rotate(Z_AXIS, initAngle, &showPoints[i], &rotOrigin );
    scale(-300.0);
    transUnit = -130;
    translate(X_AXIS);
    transUnit = -100;
    translate(Y_AXIS);
    transUnit = 25;

    is_vert_slice = 0;
    is_colp_slice = 0;
    numSegs = 0;
    numCurves = 0;
    pickSlc = pickPnt[0] = pickPnt[1] = -1;

    # ifdef SUN_VERSION
    display();
    # else
    /* routine for displaying the projected points on the screen,
    to be added.
    */
    # endif
}

```

```

#include "disp.h"
#include "interpret.h"

get_region();
selectslice();
slicemove();
slicesave();
currentslice();
maxslice();
minslice();
center();
most();
pointmove();
pointsave();
currentpoint();
quit();
help();
circum();
surfacedist();
directdist();
listnames();
runfile();

int slice_lookup();
int point_lookup();
int lookup();
char get_char();

init_fsa()
{
    int i;

    for( i=0; i<STACK_SIZE; i++){
        strcpy(slice_stack[i].name, "");
        strcpy(point_stack[i].name, "");
    }

    current_state.st = -1;
    current_state.region = current_state.slice = current_state.point = -1;

    /* command: region area */
    strcpy(reserved[0].word, "region");
    reserved[0].type = CMD;
    reserved[0].value = 0;
    fsa[0].next_state = STATE_REGION;
    fsa[0].parameter = INT;
    fsa[0].func = get_region;

    /* command: selectslice percent */
    strcpy(reserved[1].word, "selectslice");
    reserved[1].type = CMD;
    reserved[1].value = 1;
    fsa[1].next_state = STATE_SLICE;
    fsa[1].parameter = DOUBLE;
    fsa[1].func = selectslice;

    /* command: slicemove direction distance */
    strcpy(reserved[2].word, "slicemove");
    reserved[2].type = CMD;
    reserved[2].value = 2;
    fsa[2].next_state = STATE_SLICE;
    fsa[2].parameter = INT_DOUBLE;
    fsa[2].func = slicemove;

    /* command: slicesave name */
    strcpy(reserved[3].word, "slicesave");
    reserved[3].type = CMD;
    reserved[3].value = 3;
    fsa[3].next_state = STATE_SLICE;
    fsa[3].parameter = STRING;
    fsa[3].func = slicesave;

    /* command: maxslice slicel slice2 */
    strcpy(reserved[4].word, "maxslice");
    reserved[4].type = CMD;
    reserved[4].value = 4;
    fsa[4].next_state = STATE_SLICE;
    fsa[4].parameter = STRING_STRING;
    fsa[4].func = maxslice;

    /* command: minslice slicel slice2 */
    strcpy(reserved[5].word, "minslice");
    reserved[5].type = CMD;
    reserved[5].value = 5;
    fsa[5].next_state = STATE_SLICE;
    fsa[5].parameter = STRING_STRING;
    fsa[5].func = minslice;

    /* command: center position */
    strcpy(reserved[6].word, "center");
    reserved[6].type = CMD;
    reserved[6].value = 6;
    fsa[6].next_state = STATE_POINT;
    fsa[6].parameter = INT;
    fsa[6].func = center;

    /* command: most position */
    strcpy(reserved[7].word, "most");
    reserved[7].type = CMD;
    reserved[7].value = 7;
    fsa[7].next_state = STATE_POINT;
    fsa[7].parameter = INT;
    fsa[7].func = most;

    /* command: pointmove direction distance */
    strcpy(reserved[8].word, "pointmove");
    reserved[8].type = CMD;
    reserved[8].value = 8;
    fsa[8].next_state = STATE_POINT;
    fsa[8].parameter = INT_DOUBLE;
    fsa[8].func = pointmove;

    /* command: pointsave name */
    strcpy(reserved[9].word, "pointsave");
    reserved[9].type = CMD;
    reserved[9].value = 9;
    fsa[9].next_state = STATE_POINT;
    fsa[9].parameter = STRING;
    fsa[9].func = pointsave;

    /* command: quit */
    strcpy(reserved[10].word, "quit");
    reserved[10].type = CMD;
    reserved[10].value = 10;
    fsa[10].next_state = STATE_REGION;
    fsa[10].parameter = NONE;
    fsa[10].func = NULL;

```

```

/* constant: torso, value 0 */
strcpy(reserved[11].word, "torso");
reserved[11].type = CONST;
reserved[11].value = 0;

/* constant: larm, value 1 */
strcpy(reserved[12].word, "larm");
reserved[12].type = CONST;
reserved[12].value = 1;

/* constant: rarm, value 2 */
strcpy(reserved[13].word, "rarm");
reserved[13].type = CONST;
reserved[13].value = 2;

/* constant: lleg, value 3 */
strcpy(reserved[14].word, "lleg");
reserved[14].type = CONST;
reserved[14].value = 3;

/* constant: rleg, value 4 */
strcpy(reserved[15].word, "rleg");
reserved[15].type = CONST;
reserved[15].value = 4;

/* constant: upbody, value 5 */
strcpy(reserved[16].word, "upbody");
reserved[16].type = CONST;
reserved[16].value = 5;

/* constant: all, value 6 */
strcpy(reserved[17].word, "all");
reserved[17].type = CONST;
reserved[17].value = 6;

/* constant: up, value 0 */
strcpy(reserved[18].word, "up");
reserved[18].type = CONST;
reserved[18].value = 0;

/* constant: down, value 1 */
strcpy(reserved[19].word, "down");
reserved[19].type = CONST;
reserved[19].value = 1;

/* constant: clock, value 2 */
strcpy(reserved[20].word, "clock");
reserved[20].type = CONST;
reserved[20].value = 2;

/* constant: counterclock, value 3 */
strcpy(reserved[21].word, "counterclock");
reserved[21].type = CONST;
reserved[21].value = 3;

/* constant: front, value 0 */
strcpy(reserved[22].word, "front");
reserved[22].type = CONST;
reserved[22].value = 0;

/* constant: back, value 1 */
strcpy(reserved[23].word, "back");
reserved[23].type = CONST;
reserved[23].value = 1;

/* constant: left, value 2 */
strcpy(reserved[24].word, "left");
reserved[24].type = CONST;
reserved[24].value = 2;

/* constant: right, value 3 */
strcpy(reserved[25].word, "right");
reserved[25].type = CONST;
reserved[25].value = 3;

/* command: currentpoint name */
strcpy(reserved[26].word, "currentpoint");
reserved[26].type = CMD;
reserved[26].value = 11;
fsa[11].next_state = STATE_POINT;
fsa[11].parameter = STRING;
fsa[11].func = currentpoint;

/* command: currentslice name */
strcpy(reserved[27].word, "currentslice");
reserved[27].type = CMD;
reserved[27].value = 12;
fsa[12].next_state = STATE_POINT;
fsa[12].parameter = STRING;
fsa[12].func = currentslice;

/* command: circum */
strcpy(reserved[28].word, "circum");
reserved[28].type = CMD;
reserved[28].value = 13;
fsa[13].next_state = STATE_UNCHANGE;
fsa[13].parameter = NONE;
fsa[13].func = circum;

/* command: surfacedist p1 p2 */
strcpy(reserved[29].word, "surfacedist");
reserved[29].type = CMD;
reserved[29].value = 14;
fsa[14].next_state = STATE_UNCHANGE;
fsa[14].parameter = MULTI_STRING;
fsa[14].func = surfacedist;

/* command: directdist p1 p2 */
strcpy(reserved[30].word, "directdist");
reserved[30].type = CMD;
reserved[30].value = 15;
fsa[15].next_state = STATE_UNCHANGE;
fsa[15].parameter = MULTI_STRING;
fsa[15].func = directdist;

/* command: help */
strcpy(reserved[31].word, "help");
reserved[31].type = CMD;
reserved[31].value = 16;
fsa[16].next_state = STATE_UNCHANGE;
fsa[16].parameter = NONE;
fsa[16].func = help;

/* command: listnames */
strcpy(reserved[32].word, "listnames");
reserved[32].type = CMD;
reserved[32].value = 17;
fsa[17].next_state = STATE_UNCHANGE;

```

```

    fsa[17].parameter = NONE;
    fsa[17].func = listnames;

    /* command: runfile */
    strcpy(reserved[32].word, "runfile");
    reserved[32].type = CMD;
    reserved[32].value = 17;
    fsa[17].next_state = STATE_UNCHANGE;
    fsa[17].parameter = STRING;
    fsa[17].func = runfile;
}

help()
{
    printf("The following commands are available:\n\n");
    printf("region <area>\n area can be TORSO, UPBODY, LARM, RARM, LLEG, RLEG or AL\n\n");
    printf("selectslice <percentage>\n");
    printf("slicemove <direction> <distance>\n direction can be UP or DOWN.\n");
    printf("slicesave <name>\n name is a string started with a character, it should be provided by the user.\n");
    printf("currentslice <name>\n");
    printf("center <position>\n position can be FRONT, BACK, LEFT or RIGHT.\n");
    printf("most <position>\n position can be FRONT, BACK, LEFT or RIGHT.\n");
    printf("pointmove <direction> <distance>\n direction can be UP, DOWN, CLOCK or COUNTERCLOCK.\n");
    printf("pointsave <name>\n name is a string started with a character, it should be provided by the user.\n");
    printf("currentpoint <name>\n");
    printf("circum\n");
    printf("directdist <p1> <p2> ... <pn>\n p1, p2, ..., pn (n<=8) are two names provided by the user.\n");
    printf("surfacedist <p1> <p2> ... <pn>\n p1, p2, ..., pn (n<=8) are two names provided by the user.\n");
    printf("listnames\n");
    printf("runfile <filename>.\n");
    printf("quit\n");
    printf("help\n");
}

get_region(reg)
int reg;
{
    switch( reg ){
        case TORSO:
            region_torso();
            break;
        case LARM:
            region_arm(L);
            break;
        case RARM:
            region_arm(R);
            break;
        case LLEG:
            region_leg(L);
            break;
        case RLEG:
            region_leg(R);
            break;
        case UPBODY:
            region_up_torso();
            break;
        case ALL:
            region_all();
            break;
    }
    showregion(reg);
}

slicemove(direction, distance)
int direction;
double distance;
{
    if( current_state.slice == -1 ){
        printf("No current slice available. Select a slice first.\n");
        return;
    }
    if( direction==UP )
        current_state.slice = s_moveup(current_state.slice, distance);
    else if( direction==DOWN )
        current_state.slice = s_movedown(current_state.slice, distance);
    showslice(current_state.slice);
}

slicesave(str)
char *str;
{
    if( current_state.slice == -1 ){
        printf("No current slice available. Select a slice first.\n");
        return;
    }
    slice_stk_ptr = (slice_stk_ptr+1) % STACK_SIZE;
    strcpy(slice_stack[slice_stk_ptr].name, str);
    slice_stack[slice_stk_ptr].slice = current_state.slice;
    slice_stack[slice_stk_ptr].region = current_state.region;
    printf("slice %d saved to %s\n", current_state.slice, str);
}

currentslice(str)
char *str;
{
    int entry;

    if( slice_lookup(str, &entry) ){
        current_state.point = -1;
        current_state.slice = slice_stack[entry].slice;
        current_state.region = slice_stack[entry].region;
        printf("slice %d is %s\n", current_state.slice, str);
        get_region(current_state.region);
    }
    else
        showslice(current_state.slice);
}

int slice_lookup(s, entry)
char *s;
int *entry;
{
    int p;

    for( p=STACK_SIZE-1; p>=0; p-- )
        if( strcmp(slice_stack[p].name, s) == 0 ){
            *entry = p;
            return 1;
        }
    printf("Slice string %s not found.\n", s);
}

```



```

    return(0);

    maxslice(slice1, slice2)
    char *slice1, *slice2;
    {
        int s1, s2;

        if( !slice_lookup(slice1, &s1) || !slice_lookup(slice2, &s2) )
            return;
        s1 = slice_stack[s1].slice;
        s2 = slice_stack[s2].slice;
        current_state.slice = s_max(s1, s2);
        showslice(current_state.slice);
    }

    minslice(slice1, slice2)
    char *slice1, *slice2;
    {
        int s1, s2;

        if( !slice_lookup(slice1, &s1) || !slice_lookup(slice2, &s2) )
            return;
        s1 = slice_stack[s1].slice;
        s2 = slice_stack[s2].slice;
        current_state.slice = s_min(s1, s2);
        showslice(current_state.slice);
    }

    center(position)
    int position;
    {
        if( current_state.slice == -1 ){
            printf("No current slice available. Select a slice first.\n");
            return;
        }
        switch( position ){
            case FRONT:
                current_state.point = centerfront(current_state.slice);
                break;
            case BACK:
                current_state.point = centerback(current_state.slice);
                break;
            case LEFT:
                current_state.point = centerleft(current_state.slice);
                break;
            case RIGHT:
                current_state.point = centerright(current_state.slice);
                break;
            default:
                printf("Not a valid position.\n");
                return;
        }
        showpoint(current_state.point);
    }

    most(position)
    int position;
    {
        if( current_state.slice == -1 ){
            printf("No current slice available. Select a slice first.\n");
            return;
        }
        switch( position ){
            case FRONT:
                current_state.point = foremost(current_state.slice);
                break;
            case BACK:
                current_state.point = hindmost(current_state.slice);
                break;
            case LEFT:
                current_state.point = leftmost(current_state.slice);
                break;
            case RIGHT:
                current_state.point = rightmost(current_state.slice);
                break;
            default:
                printf("Not a valid position.\n");
                return;
        }
        showpoint(current_state.point);
    }

    pointmove(direction, distance)
    int direction;
    double distance;
    {
        if( current_state.point == -1 ){
            printf("No current point available. Select a point first.\n");
            return;
        }
        switch( direction ){
            case UP:
                current_state.point = p_moveup(current_state.slice, current_state.point,
                    distance);
                printf("slice %d point %d\n", current_state.slice, current_state.point);
                break;
            case DOWN:
                current_state.point = p_movedown(current_state.slice, current_state.point,
                    distance);
                printf("slice %d point %d\n", current_state.slice, current_state.point);
                break;
            case CLOCK:
                current_state.point = p_clockwise(current_state.slice, current_state.point,
                    t, distance);
                printf("slice %d point %d\n", current_state.slice, current_state.point);
                break;
            case COUNTERCLOCK:
                current_state.point = p_counter_clockwise(current_state.slice, current_state
                    .point, distance);
                printf("slice %d point %d\n", current_state.slice, current_state.point);
                break;
            default:
                printf("Not a valid direction.\n");
                return;
        }
        showpoint(current_state.point);
    }

    pointsave(str)
    char *str;
    {
        if( current_state.point == -1 ){
            printf("No current point available. Select a point first.\n");
            return;
        }
        point_stk_ptr = (point_stk_ptr+1) % STACK_SIZE;
        strcpy(point_stack[point_stk_ptr].name, str);
    }

```

```

    point_stack[point_stk_ptr].point = current_state.point;
    point_stack[point_stk_ptr].slice = current_state.slice;
    point_stack[point_stk_ptr].region = current_state.region;
    printf("Point %d saved to %s\n", current_state.point, str);
}

currentpoint(str)
char *str;
{
    int entry;

    if( !point_lookup(str, &entry) ){
        current_state.point = point_stack[entry].point;
        current_state.slice = point_stack[entry].slice;
        current_state.region = point_stack[entry].region;
        get_region(current_state.region);
    }
    else return;
    showpoint(current_state.point);

    int point_lookup(s, entry)
    char *s;
    int *entry;
    {
        int p;

        for( p=STACK_SIZE-1; p>=0; p-- )
            if( strcmp(point_stack[p].name, s) == 0 ){
                *entry = p;
                return 1;
            }
        printf("Point name %s not found.\n", s);
        return(0);
    }

    circum()
    {
        double d;

        if( current_state.slice == -1 ){
            printf("No current slice. Select a slice first.\n");
            return;
        }
        d = lengthInInches(circumference(current_state.slice));
        printf("Circumference of current slice %d is %f inches\n", current_state.slice,
d);
    }

    /* the routine is changed to multiple points. */
    directdist(str1, str2)
    char *str1, *str2;
    {
        int p1, p2;

        if( !point_lookup(str1, &p1) ){
            printf("Point %s not found.\n", str1);
            return;
        }
        else if( !point_lookup(str2, &p2) ){
            printf("Point %s not found.\n", str2);
            return;
        }
        p1 = point_stack[p1].point;
        p2 = point_stack[p2].point;
        printf("Direct distance between %s and %s is %f inches.\n", str1, str2, lengthInInches(surface_dist(p1, p2)) );
    }

    p1 = point_stack[p1].point;
    p2 = point_stack[p2].point;
    printf("Direct distance between %s and %s is %f inches.\n", str1, str2, lengthInInches(direct_dist(p1, p2)) );
}

double dist[MULT_POINT_NUM], total_dist;
int i, j;
int index[MULT_POINT_NUM];

if( !strcmp(str[i], "\0") ){
    printf("Must be more than one point.\n");
    return;
}
if( !point_lookup(str[0], &index[0]) ){
    printf("Point %s not found.\n", str[0]);
    return;
}
index[0] = point_stack[index[0]].point;
i = 1;
total_dist = 0.0;
display();
while( strcmp(str[i], "\0") ){
    if( !point_lookup(str[i], &index[i]) ){
        printf("Point %s not found.\n", str[i]);
        return;
    }
    index[i] = point_stack[index[i]].point;
    dist[i-1] = lengthInInches(direct_dist(index[i-1], index[i]));
    total_dist += dist[i-1];
    /*display();*/
    highlight_line(index[i-1], index[i]);
    /*for( j=0; j<i; j++ ) highlight_line(index[i-1], index[j]);*/
    printf("Direct distance between %s and %s is %f inches.\n", str[i-1], str[i], dist[i-1]);
    i++;
}
printf("Total distance is %f inches.\n", total_dist);

/* the routine is changed to multiple points. */
surfacedist(str1, str2)
char *str1, *str2;
{
    int p1, p2;

    if( !point_lookup(str1, &p1) ){
        printf("Point %s not found.\n", str1);
        return;
    }
    else if( !point_lookup(str2, &p2) ){
        printf("Point %s not found.\n", str2);
        return;
    }
    p1 = point_stack[p1].point;
    p2 = point_stack[p2].point;
    printf("Direct distance between %s and %s is %f inches.\n", str1, str2, lengthInInches(surface_dist(p1, p2)) );
}

```

```

*/
surfacedist(str)
char **str;
{
    double dist[MULT_POINT_NUM], total_dist;
    int i, j;
    curve_dis curves[MULT_POINT_NUM];
    int index[MULT_POINT_NUM];

    if( !strcmp(str[1], "0") ){
        printf("Must be more than one point.\n");
        return;
    }
    if( !point_lookup(str[0], &index[0]) ){
        printf("Point %s not found.\n", str[0]);
        return;
    }
    index[0] = point_stack[index[0]].point;
    i = 1;
    total_dist = 0.0;
    while( strcmp(str[i], "0") ){
        if( !point_lookup(str[i], &index[i]) ){
            printf("Point %s not found.\n", str[i]);
            return;
        }
        index[i] = point_stack[index[i]].point;
        dist[i-1] = lengthInches(surface_dist(index[i-1], index[i], &curves[i]
        -1) ));
        total_dist += dist[i-1];
        display();
        for( j=0; j<i; j++ ) highlight_curve(curves[j]);
        printf("Surface distance between %s and %s is %f inches.\n", str[i-1], s
        tr[i], dist[i-1]);
        i++;
    }
    printf("Total distance is %f inches.\n", total_dist);
}

listnames()
{
    int i;

    printf("Named slices:\n");
    for(i=0; i<STACK_SIZE; i++)
        if( strcmp(slice_stack[i].name, " ") )
            printf("%s: %d\n", slice_stack[i].name, slice_stack[i].slice);
    printf("Named points:\n");
    for(i=0; i<STACK_SIZE; i++)
        if( strcmp(point_stack[i].name, " ") )
            printf("%s: %d\n", point_stack[i].name, point_stack[i].point);
}

runfile(str)
char *str;
{
    fp = fopen(str, "r");
    if( fp == NULL )
        fprintf(stderr, "cannot read the file\n");
    else mode = BATCH;
}

interpret()
{
    int i1, i2;
    double d1;
    char str[MAX_STRING_LENGTH];
    int cmd_num;
    int count;
    char *(str_array[8]);

    mode = INTERACTIVE;
    for( count=0; count<8; count++ )
        str_array[count] = (char *) malloc( MAX_STRING_LENGTH*sizeof(char) );
    while(1) {
        printf("3D> ");
        cmd_num = get_command();
        if( cmd_num == -1 ){
            printf("command not found\n");
            continue;
        }
        if( cmd_num == 10 ) return; /* command quit */
        if( fsa[cmd_num].next_state )
            current_state.st = fsa[cmd_num].next_state;
        switch (fsa[cmd_num].parameter) {
            case NONE:
                (fsa[cmd_num].func)();
                break;
            case INT:
                if( !get_int(&i1) ) continue;
                (fsa[cmd_num].func)(i1);
                break;
            case DOUBLE:
                if( !get_double(&d1) ) continue;
                (fsa[cmd_num].func)(d1);
                break;
            case INT_INT:
                if( !get_int(&i1) ) continue;
                if( !get_int(&i2) ) continue;
                (fsa[cmd_num].func)(i1, i2);
                break;
            case INT_DOUBLE:
                if( !get_int(&i1) ) continue;
                if( !get_double(&d1) ) continue;
                (fsa[cmd_num].func)(i1, d1);
                break;
            case STRING:
                if( !get_string() ) continue;
                (fsa[cmd_num].func)(token_str);
                break;
            case STRING_STRING:
                if( !get_string() ) continue;
                strcpy(str, token_str);
                if( !get_string() ) continue;
                (fsa[cmd_num].func)(str, token_str);
                break;
            case MULTI_STRING:
                for( count=0; count<8; count++ )
                    strcpy( str_array[count], "\0" );
                count = -1;
                while( get_string() != TYPE_LAST_STRING && count<8 )
                    strcpy(str_array[++count], token_str);
                strcpy(str_array[++count], token_str);
                (fsa[cmd_num].func)(str_array);
                break;
        }
    }
}

```



```
token_type = TYPE_RETURN;
token_str[0] = '\n';
return;
}

else (
    print_error(0);
    return;
)

char get_char()
{
    int t;

    if( INTERACTIVE == mode ) t = getchar();
    else(
        t = getc(fp);
        if( t == EOF )
        {
            mode = INTERACTIVE;
            printf("\nfile execution finished\n");
            printf("3D> ");
            return getchar();
        }
        printf("%c", t);
    )
    return t;
}

print_error(num)
int num;
{
    switch( num ){
        case 0:
            printf("\nCommand not valid.\n");
            break;
        case 1:
            printf("\nMust be an integer.\n");
            break;
        case 2:
            printf("\nMust be a number.\n");
            break;
        case 3:
            printf("\nMust be a string.\n");
            break;
        case 4:
            printf("\nConstant string not found.\n");
            break;
    }
}
```

```

#include "manips.h"

/*****
Translation around X direction.
*****/
void
transX()
{
    translate(X_AXIS);
    display();
}

/*****
Translation around Y direction.
*****/
void
transY()
{
    translate(Y_AXIS);
    display();
}

/*****
Translation around Z direction.
*****/
void
transZ()
{
    translate(Z_AXIS);
    display();
}

/*****
Quit procedure.
*****/
void
quit_proc()
{
    void
    chSpeed(sign)
    int sign;
    {
        if (!sign) speedUnit *= 2;
        else speedUnit /= 2;
    }
}

/*****
Rotation around X axis.
*****/
void
rotX(rUnit)
double rUnit;
{
    int i;
    for( i=0; i<numPoints; i++ )
        rotate(X_AXIS, rUnit, &showPoints[i], &rotOrigin);
    display();
}

/*****
Rotation around Y axis.
*****/
void
rotY(rUnit)
double rUnit;
{
    int i;
    for( i=0; i<numPoints; i++ )
        rotate(Y_AXIS, rUnit, &showPoints[i], &rotOrigin);
    display();
}

/*****
Rotation around Z axis.
*****/
void
rotZ(rUnit)
double rUnit;
{
    int i;
    for( i=0; i<numPoints; i++ )
        rotate(Z_AXIS, rUnit, &showPoints[i], &rotOrigin);
    display();
}

/*****
Quit procedure.
*****/
void
quit_proc()
{
    void
    chSpeed(sign)
    int sign;
    {
        if (!sign) speedUnit *= 2;
        else speedUnit /= 2;
    }
}

```

```

#include "meas_menu.h"

char *methods[] = { "Multipoint", "Radial Measurement", "Circumference", "Language"
};
Window measMenu_win, meas_wins[ MAX_MEASMENTS+1 ];
GC measMenu_gc, meas_gcs[ MAX_MEASMENTS+1 ];
XColor fg_col, bg_col;
int measCnt;
int cur_meas=5;

Initialize_measMenu()
{
    XFontStruct *fontStr;
    int i;

    measCnt = MAX_MEASMENTS;
    create_meas_menu_windows(&measMenu_win, &measMenu_gc, meas_wins, meas_gcs, measCnt);

    fontStr = XLoadQueryFont( ncdisplay, "-adobe-courier-bold-r-normal--0-0-75-75-m-0-iso8859-1" );
    if (fontStr != 0)
        for (i=0; i<measCnt; i++)
            XSetFont( ncdisplay, meas_gcs[i], fontStr->fid);
}

refreshmeasMenu()
{
    int i;

    for (i=0; i<measCnt; i++)
        XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
    XDrawString(ncdisplay, meas_wins[measCnt], meas_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("Done"));
}

mapmeasMenu(x,y)
int x,y;
{
    int i;
    XEvent df_event;
    XWindowAttributes winAttrs;

    XMapRaised(ncdisplay, measMenu_win);
    for (i=0; i<measCnt+1; i++)
        XMapRaised(ncdisplay, meas_wins[i]);

    XGetWindowAttributes( ncdisplay, meas_wins[measCnt], &winAttrs );
    if (winAttrs.map_state != 2)
        XWindowEvent( ncdisplay, meas_wins[measCnt], ExposureMask, &df_event );

    redrawmeasMenu();
}

redrawmeasMenu()
{
    int i;

    for (i=0; i<measCnt; i++)
        XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
}

XDrawString(ncdisplay, meas_wins[measCnt], meas_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("Done"));
XFlush( ncdisplay );
}

measMenu()
{
    int i;
    char filename[MAX_FILE_NAME];

    switch (nccurrent.type)
    {
        case Expose :
            for (i=0; i<measCnt+1; i++)
                if (nccurrent.expose.window == meas_wins[i]) break;
            if ( i < measCnt+1 ) {
                for (i=0; i<measCnt; i++)
                    XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
                XDrawString(ncdisplay, meas_wins[measCnt], meas_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("Done"));
            }
            break;

        case MapNotify :
            if (nccurrent.xmap.window == measMenu_win)
                for (i=0; i<measCnt; i++) {
                    XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
                }
            XDrawString(ncdisplay, meas_wins[measCnt], meas_gcs[0], STR_LOC_X, STR_LOC_Y, "None", strlen("None"));
            break;

        case ButtonPress :
            for (i=0; i<measCnt; i++)
                if (nccurrent.xbutton.subwindow == meas_wins[i]) break;
            if ( i == measCnt ) {
                XUnmapWindow( ncdisplay, measMenu_win );
                XFlush( ncdisplay );
            }
            else {
                XSetForeground( ncdisplay, meas_gcs[i], bg_col.pixel );
                XSetWindowBackground( ncdisplay, meas_wins[i], fg_col.pixel );
                XClearWindow( ncdisplay, meas_wins[i] );
                XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
                XFlush( ncdisplay );
            }
            manage_meas_menu(i);

            XSetForeground( ncdisplay, meas_gcs[i], fg_col.pixel );
            XSetWindowBackground( ncdisplay, meas_wins[i], bg_col.pixel );
            XClearWindow( ncdisplay, meas_wins[i] );
            XDrawString(ncdisplay, meas_wins[i], meas_gcs[i], STR_LOC_X, STR_LOC_Y, methods[i], strlen(methods[i]));
            XFlush( ncdisplay );
        }
    }
}

```

```

meas_hints[i].height, 0, depth, InputOutput.CopyFromParent, dfile_win_mask,
&dfile_win_attrs);
    )
    /* Create a graphics context */
    *gc = XCreateGC ( ncdisplay, *win, 0, 0);
    XSetBackground (ncdisplay, *gc, bg_col.pixel);
    XSetWindowBackground (ncdisplay, *win, bg_col.pixel);
    XSetForeground (ncdisplay, *gc, fg_col.pixel);
    XSelectInput (ncdisplay, *win, ButtonPressMask | ExposureMask | StructureNotifyMask
sk);

    for (i=0; i<num_files+1; i++) {
        meas_gcs[i] = XCreateGC ( ncdisplay, meas_wins[i], 0, 0);
        XSetBackground (ncdisplay, meas_gcs[i], bg_col.pixel);
        XSetWindowBackground (ncdisplay, meas_wins[i], bg_col.pixel);
        XSetForeground (ncdisplay, meas_gcs[i], fg_col.pixel);
        XSelectInput (ncdisplay, meas_wins[i], ExposureMask | StructureNotifyMask
);
    }

    )

    manage_meas_menu(item_no)
    int
    item_no;
    {
        switch (item_no)
        {
            case 0: measSequence();
                    break;

            case 1: measRadial();
                    break;

            case 2: measSlice();
                    break;

            case 3: /*language();*/
                    reset();
                    init_fsa();
                    interpret();
                    break;

            default: break;
        }
    }

    create_meas_menu_windows(win_gc, meas_wins, meas_gcs, num_files)
Window *win, *meas_wins;
GC *gc, *meas_gcs;
int num_files;
{
    XSetWindowAttributes dfile_win_attrs;
    unsigned long dfile_win_mask;
    XSizeHints dfile_hint, meas_hints[MAX_MEASMTS];
    int i, depth;

    depth = XDisplayPlanes( ncdisplay, ncscreen);

    bg_col.red = meas_BG_RED;
    bg_col.blue = meas_BG_BLUE;
    bg_col.green = meas_BG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &bg_col);

    fg_col.red = meas_FG_RED;
    fg_col.blue = meas_FG_BLUE;
    fg_col.green = meas_FG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &fg_col);

    dfile_win_attrs.border_pixel = fg_col.pixel;
    dfile_win_attrs.background_pixel = bg_col.pixel;
    dfile_win_attrs.override_redirect = True;

    dfile_win_mask = ( CWBorderPixel | CWOverrideRedirect );

    /* set the window parameters for dfile window */
    dfile_hint.x = 800; dfile_hint.y = 450;
    dfile_hint.width = 150; dfile_hint.height = (num_files+1) * FN_WIDTH;
    dfile_hint.flags = USPosition | PSize;

    /* Create the window with the above given specifications */
    *win = XCreateSimpleWindow ( ncdisplay, DefaultRootWindow ( ncdisplay ), dfile_h
int.x, dfile_hint.y,
                                dfile_hint.width, dfile_hint.height, 5, fg_col.pixel, bg
_col.pixel);
    XSetStandardProperties( ncdisplay, *win, "MEASUREMENTS", "MEASUREMENTS", None, 0 )
, 0, &dfile_hint );

    /* set the window parameters for dfile sub windows */
    for (i=0; i<num_files+1; i++) {
        meas_hints[i].x = 0;
        meas_hints[i].y = i*FN_WIDTH;
        meas_hints[i].width = 150; meas_hints[i].height = FN_WIDTH;
        meas_hints[i].flags = USPosition | PSize;

        meas_wins[i] = XCreateWindow(ncdisplay, *win, meas_hints[i].x, meas_hints[i].y, me
as_hints[i].width,

```



```

#include "measure.h"

double realSliceLen();
double distance();
double sliceLen();
double pointLen();
double lengthInInches();
double inchInLength();
double yz_dist();

/* Slice circumference in inches */
double sliceCircumference(sliceNo)
int sliceNo;
{
    return realSliceLen(sliceNo)/stdChest*stdInchChest;
}

/* Distance of inches between 2 points */
double pointDist(p1,p2)
int p1, p2;
{
    return distance(allPoints[p1],allPoints[p2])/stdChest*stdInchChest;
}

/* Slice circumference in world coordinate */
double realSliceLen(sliceNo)
int sliceNo;
{
    int i, sliceNumPoints;
    double length=0.0;

    sliceNumPoints = allSlices[sliceNo+1] - allSlices[sliceNo] + 1;
    for (i=allSlices[sliceNo]; i<allSlices[sliceNo+1]-1; i++)
        length += distance(allPoints[i],allPoints[i+1]);
    length += distance(allPoints[i],allPoints[allSlices[sliceNo]]);

    return(length);
}

/* Distance of 2 points in world coordinate */
double distance(p1,p2)
point3 p1,p2;
{
    return sqrt( (p2.x-p1.x)*(p2.x-p1.x) + (p2.y-p1.y)*(p2.y-p1.y) + (p2.z-p1.z)*(p2
.z-p1.z) );
}

/* Conversion between inches and world coordinate standard */
set_meas_standard()
{
    /* stdChest = realSliceLen(chest); */
    stdChest = 975.0;
    stdInchChest = 38.0;
}

/* Convert from world coordinate to inch */
double lengthInInches(realLen)
double realLen;
{
    return realLen/stdChest*stdInchChest;
}

/* Convert from inch to world coordinate */
double inchInLength(inch)

```

```

/*****
/*
/* FILENAME : menu.c
/*
/* This file contains all the routines used to display the Menu items,
/* collect the input from them etc.
/*
/* *****/

# include <X11/Xlib.h>
# include <X11/Xutil.h>

# include <stdio.h>
# include "menu.h"

struct menu_item
int
struct name
int
Items[MAX_BUTTONS];
Separators[MAX_SEPARATORS];
Names[MAX_NAMES];
butt_ctr=0, sepr_ctr=0, name_ctr=0;;

XColor butCol1,butCol2,butCol3;
extern Display *ncdisplay;
extern Window menu_window;
extern GC menu_gc;

Initialize_Menu()
{
    CreateMenuButton(DATA_FILES_BUTTON,10,20,50,20,"Load");
    CreateMenuButton(MEASURE_BUTTON,65,20,65,20,"Measure");
    CreateMenuButton(SLICE_BUTTON,135,20,60,20,"Slices");
    CreateMenuButton(RESET_BUTTON,200,20,50,20,"Reset");
    CreateMenuButton(QUIT_BUTTON,255,20,45,20,"Quit");

    createSeparator(55);

    createName(20,80,"ROTATE");
    CreateMenuButton(ROTATE_X_BUTTON,20,90,25,20,"X");
    CreateMenuButton(ROTATE_Y_BUTTON,55,90,25,20,"Y");
    CreateMenuButton(ROTATE_Z_BUTTON,90,90,25,20,"Z");
    createPMeter("Rotate Angle",ROTATE_MET,135,105,120,-180,180,30,'d');

    createName(20,145,"TRANSLATE");
    CreateMenuButton(TRANS_X_BUTTON,20,155,25,20,"X");
    CreateMenuButton(TRANS_Y_BUTTON,55,155,25,20,"Y");
    CreateMenuButton(TRANS_Z_BUTTON,90,155,25,20,"Z");
    createPMeter("Translation Offset",TRANS_MET,135,170,120,-80,80,25,'p');

    CreateMenuButton(SCALE_BUTTON,20,215,60,20,"Scale");
    createPMeter("Scale Factor",SCALE_MET,135,230,120,-100,100,25,'%');

    createName(20,275,"AUTO ROTATE");
    CreateMenuButton(AUTO_X_BUTTON,20,285,25,20,"X");
    CreateMenuButton(AUTO_Y_BUTTON,55,285,25,20,"Y");
    CreateMenuButton(AUTO_Z_BUTTON,90,285,25,20,"Z");
    CreateMenuButton(AUTO_RANDOM_BUTTON,140,285,60,20,"Random");
    CreateMenuButton(AUTO_STOP_BUTTON,245,285,50,20,"Stop");
    createPMeter("Rotation Speed",AUTO_ROT_MET,135,350,120,5,100,25,'r');
}

CreateMenuButton(butt_no,x,y,width,height,title)
int butt_no;
int x,y,width,height;
char *title;
{
    XPoint region[4];

    Items[butt_ctr].butt_no = butt_no;
    Items[butt_ctr].x = x;
    Items[butt_ctr].y = y;
    Items[butt_ctr].width = width;
    Items[butt_ctr].height = height;
    strcpy(Items[butt_ctr].title,title);

    region[0].x = x; region[0].y = y;
    region[1].x = x; region[1].y = y + height;
    region[2].x = x + width; region[2].y = y + height;
    region[3].x = x + width; region[3].y = y;
    Items[butt_ctr].bounds = XPolygonRegion( region, 4, EvenOddRule );

    butt_ctr++;
}

check_menu_item(x,y,number,class)
int x,y;
int *number,*class;
{
    int i;

    for (i=0;i<butt_ctr;i++)
        if (XPointInRegion(Items[i].bounds,x,y)) {
            *number = Items[i].butt_no;
            *class = BUTTON;
            return;
        }

    for (i=0;i<pmeter_ctr;i++)
        if (XPointInRegion(pMeters[i].bounds,x,y) ||
            XPointInRegion(pMeters[i].curvalReg,x,y))
        {
            *number = pMeters[i].pMNum;
            *class = PMETER;
            return;
        }

    *number = 0;
    *class = BG_BUTTON;
}

drawButton( x,y,width,height,flag )
int x,y,width,height;
int flag;
{
    XPoint rect[4];
    unsigned long temp_fg;

    temp_fg = menu_gc->values.foreground;

    if (flag)
        XSetForeground( ncdisplay, menu_gc, butCol1.pixel );
    else

```

```

XSetForeground( ncdisplay, menu_gc, butCol2.pixel );

rect[0].x = x; rect[0].y = y;
rect[1].x = x; rect[1].y = y+height;
rect[2].x = x-2; rect[2].y = y+height+2;
rect[3].x = x-2; rect[3].y = y-2;
XFillPolygon( ncdisplay, menu_window, menu_gc, rect, 4, Convex, CoordModeOrigin
);

rect[0].x = x; rect[0].y = y;
rect[1].x = x+width; rect[1].y = y;
rect[2].x = x+width+2; rect[2].y = y-2;
rect[3].x = x-2; rect[3].y = y-2;
XFillPolygon( ncdisplay, menu_window, menu_gc, rect, 4, Convex, CoordModeOrigin
);

if (flag)
    XSetForeground( ncdisplay, menu_gc, butCol2.pixel );
else
    XSetForeground( ncdisplay, menu_gc, butCol1.pixel );

rect[0].x = x; rect[0].y = y+height;
rect[1].x = x+width; rect[1].y = y+height;
rect[2].x = x+width+2; rect[2].y = y+height+2;
rect[3].x = x-2; rect[3].y = y+height+2;
XFillPolygon( ncdisplay, menu_window, menu_gc, rect, 4, Convex, CoordModeOrigin
);

rect[0].x = x+width; rect[0].y = y;
rect[1].x = x+width; rect[1].y = y+height;
rect[2].x = x+width+2; rect[2].y = y+height+2;
rect[3].x = x+width+2; rect[3].y = y-2;
XFillPolygon( ncdisplay, menu_window, menu_gc, rect, 4, Convex, CoordModeOrigin
);

XSetForeground( ncdisplay, menu_gc, temp_fg );

redrawMenu()
{
    int i;
    for (i=0; i<butt_ctr; i++) {
        drawButton(Items[i].x, Items[i].y, Items[i].width, Items[i].height, 1);
        XDrawString( ncdisplay, menu_window, menu_gc, Items[i].x + 5,
                    Items[i].y + 15, Items[i].title, strlen(Items[i].title));
    }
    for (i=0; i<pmeter_ctr; i++)
        drawPMeter( ncdisplay, menu_window, menu_pm_gc, pMeters[i]);
    for (i=0; i<sepr_ctr; i++)
        drawSeparator( ncdisplay, menu_window, menu_gc, i );
    for (i=0; i<name_ctr; i++)
        drawName( ncdisplay, menu_window, menu_gc, i );
    fprintf(stderr, "Outside redrawMenu.\n");
}

/*
*/
}

inverse(button, flag)
int button, flag;
{
    int i, butt_no=button;
    unsigned long temp_fg;
    for (i=0; i<butt_ctr; i++)
        if ( Items[i].butt_no == butt_no ) break;

    drawButton( Items[i].x, Items[i].y, Items[i].width, Items[i].height, flag);
    XFlush( ncdisplay );
}

createSeparator(y)
int y;
{
    Separators[sepr_ctr++] = y;
}

drawSeparator( disp, win, gc, index)
Display *disp;
Window win;
GC gc;
int index;
{
    unsigned long temp_fg;
    temp_fg = menu_gc->values.foreground;
    XSetForeground( ncdisplay, menu_gc, butCol1.pixel );
    XDrawLine( ncdisplay, menu_window, menu_gc, 0, Separators[index], 920, Separators
[index] );
    XSetForeground( ncdisplay, menu_gc, butCol2.pixel );
    XDrawLine( ncdisplay, menu_window, menu_gc, 0, Separators[index]+2, 920, Separato
rs[index]+2 );
    XSetForeground( ncdisplay, menu_gc, temp_fg );
}

createName(x, y, name)
int x, y;
char *name;
{
    Names[name_ctr].x = x;
    Names[name_ctr].y = y;
    strcpy(Names[name_ctr].name, name);
    name_ctr++;
}

drawName(disp, win, gc, inx)
Display *disp;
Window win;
GC gc;
int inx;
{
    XDrawString( disp, win, gc, Names[inx].x, Names[inx].y, Names[inx].name, strlen(N
ames[inx].name) );
}

```

```

#include "multi_meas.h"
#include <stdio.h>

curve_dis  cdistances[MAX_SEGMENTS];
int         pickPnts[MAX_SEGMENTS+1][2];
int         numSegs;
double      total_seq_len;

void measSequence();
void pickAllPoints();
void computeAllPaths();

void measSequence()
{
    int i;

    for (i=0; i<MAX_SEGMENTS+1; i++)
        pickPnts[i][0] = pickPnts[i][1] = -1;
    /* to differentiate between which is measured at the latest */
    numCurves = 0;
    pickSlc = -1;

    fprintf(stderr, " Inside measSequence(). *****\n");

    numSegs = 0;
    display();
    pickAllPoints();
    computeAllPaths();
}

void pickAllPoints()
{
    XEvent pickEvent;
    int i, done, segNo;

    segNo = 0;

    done = 0;
    while (!done)
    {
        XNextEvent(ncdisplay, &pickEvent);

        switch (pickEvent.type)
        {
            case ButtonPress :
                if (pickEvent.xbutton.window != ncwindow) break;

                if (pickEvent.xbutton.button == 3) {
                    numSegs = segNo-1;
                    done = 1;
                    break;
                }

                if (pickEvent.xbutton.button == 2) {
                    segNo--;
                    if (segNo == 0) break;
                    xclearWindow(ncdisplay, ncwindow);
                    display();
                    for (i=0; i<segNo; i++)
                        highlight_point(pickPnts[i][0]);
                }
            }
        }
    }

    if (pickEvent.xbutton.button == 1) {
        pickPnts[segNo][0] = pickPoint(pickEvent.xbutton.x, pickEvent.xbutton.y);
        if (segNo != 0)
            pickPnts[segNo-1][1] = pickPnts[segNo][0];

        highlight_point(pickPnts[segNo][0]);
        segNo++;
    }
    break;
    default :
        break;
}

void computeAllPaths()
{
    int i;
    double cum_dist;

    cum_dist = 0.0;
    for (i=0; i<numSegs; i++)
    {
        comp_path( pickPnts[i][0], pickPnts[i][1], &cDistances[i] );
        cum_dist += cDistances[i].ds;
        highlight_curve(cDistances[i]);
    }

    total_seq_len = cum_dist;
    display();
    fprintf(stderr, "total length = %f\n", total_seq_len);
}

```

```

#include "disp.h"

extern point3 *showPoints;
extern int numPoints;
extern int allSlices[MAX_POINT_NUM];
extern int numSlices;
extern point3 rotOrigin;

#define PI 3.1415926535897932352626

int pickPoint();
int pickSlice();
double dist2D();

extern int *pickPnts;
int pickPnt[2]=(-1,-1);
int pickSlc=-1;

/*****
 * This routine returns the 3D point entry
 * whose 2D projection is closest to the
 * given 2D position.
 *****/
int pickPoint(x,y)
int x,y;
{
    point3 p3;
    point2 p2, pnt2;
    int i;
    int pointNum;
    double dist, dist2;

    pointNum = -1;
    p2.x = x;
    p2.y = y;
    projection(&(showPoints[0]), &pnt2, 0);
    dist = dist2D(pnt2, p2);
    for( i=1; i<numPoints; i++ ) {
        projection(&(showPoints[i]), &pnt2, i);
        dist2 = dist2D(pnt2, p2);
        if( dist2<dist && dist2<5 ) {
            dist = dist2;
            pointNum = i;
        }
    }
    return(pointNum);
}

/*****
 * This routine converts a 2D screen
 * position to the 2D display
 * coordinate.
 *****/
srcnTo2D(x,y,pt2)
int x,y;
point2 *pt2;
{
    point2 p2;

    projection(&rotOrigin, &p2, 1);
    pt2->x = (double)x - p2.x;
    pt2->y = (double)y - p2.y;
    rotate2D(pt2,(0.5)*PI);
}

i = 0;
end = 0;
while( end!=1 && i!=numSlices ) {
    if( allSlices[i]<=pointNo && allSlices[i+1]>pointNo )
        else i++;
    if( i==numSlices ) return(-1);
    else return(i);
}

/*****
 * This routine calculates 2D distance
 * between two 2D points.
 *****/
double dist2D(p1,p2)
point2 p1,p2;
{
    return(sqrt((p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y)));
}

showPickPoint(x,y,no)
int x,y;
int no;
{
    pickSlc=-1;
    pickPnt[no-1] = pickPoint(x, y);
    display();
    if( pickPnt[no-1]==-1 ) pickPnt[0]=pickPnt[1]=-1;
    /*highlight_line(pickPnt[0],pickPnt[1]);*/
    if( pickPnt[0]>=0 && pickPnt[0]<numPoints && pickPnt[1]>=0 && pickPnt[1]<numPoints )
        s )
            comp_path(pickPnt[0], pickPnt[1], 0);
            highlight_curve();
        }
    showPickSlice(x,y)
    int x,y;
    {
        pickPnt[0]=pickPnt[1]=-1;
        pickSlc = pickSlice(x, y);
        display();
        /*if( pickSlc!=-1 ) highlight(pickSlc); */
    }

/*****
 * This routine converts a 2D screen
 * position to the 2D display
 * coordinate.
 *****/
srcnTo2D(x,y,pt2)
int x,y;
point2 *pt2;
{
    point2 p2;

    projection(&rotOrigin, &p2, 1);
    pt2->x = (double)x - p2.x;
    pt2->y = (double)y - p2.y;
    rotate2D(pt2,(0.5)*PI);
}

```

```
# include "pmeter.h"
# include <stdio.h>

struct pmeter pmeters[MAX_PMETERS];
int pmeterCtr=0;

createPMeter(pMName,pmNum,x,y,extent,leftVal,rightVal,initVal,unit)
char *pMName;
int pmNum;
int x,y,extent;
int leftVal,rightVal,initVal;
char unit;
{
    int i;
    XPoint region[4];
    XPoint region1[4];
    i = pmeterCtr;

    strcpy(pMeters[i].pMName,pMName);
    pMeters[i].pmNum = pmNum;
    pMeters[i].x = x;
    pMeters[i].y = y;
    pMeters[i].extent = extent;
    pMeters[i].leftVal = leftVal;
    pMeters[i].rightVal = rightVal;

    region[0].x = x; region[0].y = y-10;
    region[1].x = x; region[1].y = y + PMETER_WIDTH + 5;
    region[2].x = x + extent + 1; region[2].y = y + PMETER_WIDTH + 5;
    region[3].x = x + extent + 1; region[3].y = y;
    pMeters[i].bounds = XPolygonRegion( region, 4, EvenOddRule );

    region1[0].x = x + extent + 45; region1[0].y = y-10;
    region1[1].x = x + extent + 45; region1[1].y = y + 20;
    region1[2].x = x + extent + 8; region1[2].y = y + 20;
    region1[3].x = x + extent + 8; region1[3].y = y-10;
    pMeters[i].curvalReg = XPolygonRegion( region1, 4, EvenOddRule );

    pMeters[i].curVal = initVal;
    pMeters[i].curPosX = x + (int)((double)(initVal-leftVal)/((double)(rightVal-leftVal-
    extent)));
    pMeters[i].curPosY = y + (int)(PMEter_WIDTH/2);

    pMeters[i].unit = unit;
    pmeterCtr++;
}

drawPMeter(disp,win,gc,pMeter)
Display *disp;
Window win;
GC gc;
struct pmeter pMeter;
{
    int trix,triLy;
    char str[40];
    int namelen;
    char unit[2];

    drawMeter( disp, win, gc, pMeter );
}

drawPointer( disp, win, menu_xor_gc, pMeter.curPosX, pMeter.curPosY, pMeter.y, 0 );

drawCurVal( disp, win, gc, &pMeter, 0 );

sprintf(unit,"%c",pMeter.unit);

sprintf(str,"%20s\n",pMeter.pMName);
namelen = strlen(pMeter.pMName);
XDrawString( disp, win, gc, pMeter.x+40, pMeter.y-16, str, strlen(str) );
sprintf(str,"%d",pMeter.leftVal);
strcat(str,unit);
XDrawString( disp, win, gc, pMeter.x, pMeter.y-6, str, strlen(str) );
sprintf(str,"%d",pMeter.rightVal);
strcat(str,unit);
XDrawString( disp, win, gc, pMeter.x+(pMeter.extent-27), pMeter.y-6, str, strlen(st
r) );

XFlush ( disp );
}

drawMeter( disp, win, gc, pMeter)
Display *disp;
Window win;
GC gc;
struct pmeter pMeter;
{
    XDrawLine(disp,win,gc,pMeter.x,pMeter.y,pMeter.x,pMeter.y+PMEter_WIDTH);
    XDrawLine(disp,win,gc,pMeter.x+pMeter.extent,pMeter.y,pMeter.x+pMeter.extent,pMet
er.y+PMEter_WIDTH);

    XDrawLine(disp,win,gc,pMeter.x,pMeter.y+PMEter_WIDTH/2-2,pMeter.x+pMeter.extent,p
Meter.y+PMEter_WIDTH/2-2);
    XDrawLine(disp,win,gc,pMeter.x,pMeter.y+PMEter_WIDTH/2+2,pMeter.x+pMeter.extent,p
Meter.y+PMEter_WIDTH/2+2);

    drawPointer( disp, win, gc, x, y, flag )
    Display *disp;
    Window win;
    GC gc;
    int x,y;
    int flag;
    {
        XPoint rect[4];
        unsigned long temp_fg;

        temp_fg = gc->values.foreground;

        if (flag)
            XSetForeground( disp, gc, butCol2.pixel );
        else
            XSetForeground( disp, gc, butCol1.pixel );

        rect[0].x = x; rect[0].y = y+2;
        rect[1].x = x; rect[1].y = y;
        rect[2].x = x-4; rect[2].y = y-4;
        rect[3].x = x-8; rect[3].y = y-6;
        XFillPolygon( disp, win, gc, rect, 4, Convex, CoordModeOrigin );

        rect[0].x = x-8; rect[0].y = y-6;
    }
}
```

```

rect[1].x = x-4; rect[1].y = y-4;
rect[2].x = x+4; rect[2].y = y-4;
rect[3].x = x+8; rect[3].y = y-6;
XFillPolygon( disp, win, gc, rect, 4, Convex, CoordModeOrigin );

if (flag)
    XSetForeground( disp, gc, butCol1.pixel );
else
    XSetForeground( disp, gc, butCol2.pixel );

rect[0].x = x; rect[0].y = y+2;
rect[1].x = x; rect[1].y = y;
rect[2].x = x+4; rect[2].y = y-4;
rect[3].x = x+8; rect[3].y = y-6;
XFillPolygon( disp, win, gc, rect, 4, Convex, CoordModeOrigin );

XSetForeground( disp, gc, temp_fg );
}

drawCurVal( disp, win, gc, pMeter, flag)
Display *disp;
Window win;
GC gc;
struct pMeter *pMeter;
int flag;
{
    int x,y,width,height;
    int curVal;
    char text[20];
    char unit[2];

    x = pMeter->x + pMeter->extent + 8;
    y = pMeter->y-10;
    width = 35;
    height = 20;

    sprintf(unit,"%c",pMeter->unit);

    XClearArea( disp, win, x, y, width, height, 0 );
    XDrawRectangle( disp, win, gc, x, y, width, height );
    if (!flag)
        pMeter->curVal = (int)((double)(pMeter->curPosX-pMeter->x)/(double)pMet
er->extent)*(double)(pMeter->rightVal - pMeter->leftVal) + pMeter->leftVal;
    printf(text,"%3d",pMeter->curVal);
    strcat(text,unit);
    XDrawString( disp, win, gc, x+2, y+15, text, strlen(text) );
}

changeAngle(x,y,angle,pmNum)
int x,y;
double *angle;
int pmNum;
{
    int curVal;

    updatePointer(x,y,pmNum,&curVal);
    *angle = (double) curVal*PI/180.0;

    changeTrans(x,y,transUnit,pmNum)

```



```

    curVal = atoi(str);
    if (str_index > 4) done = 1;
    XClearArea(ncdisplay, menu_window, x+1, y+1, wid
    ,y+15, str, strlen(str));
    XDrawString(ncdisplay, menu_window, menu_pm_gc, x+2
    ,y+15, str, strlen(str));
    }
    else if ((key == XK_Delete) || (key == XK_BackSpace))
    {
        if (str_index != 0)
            str[--str_index] = '\0';
        XClearArea(ncdisplay, menu_window, x+1, y+1, wid
        ,y+15, str, strlen(str));
        XDrawString(ncdisplay, menu_window, menu_pm_gc, x+2
        ,y+15, str, strlen(str));
    }
    else if (key == XK_Return)
        done = 1;

    break;

    default :
        break;
    }

    if (curVal < pMeter->leftVal)
        curVal = pMeter->leftVal;
    if (curVal > pMeter->rightVal)
        curVal = pMeter->rightVal;

    *val = curVal;

    drawPointer(ncdisplay, menu_window, menu_xor_gc, pMeter->curPosx, pMeter->y, 0);

    pMeter->curVal = curVal;
    pMeter->curPosx = (int) (((double) (curVal - pMeter->leftVal) / (double) (pMeter->rightVal - pMeter->leftVal)) * (double) pMeter->extent) + pMeter->x;

    drawPointer(ncdisplay, menu_window, menu_xor_gc, pMeter->curPosx, pMeter->y, 0);
    drawCurVal(ncdisplay, menu_window, menu_pm_gc, pMeter, 1);

```

```

#include <stdio.h>
#include <string.h>
#include <disp.h>

extern int adjacent[2][MAX_POINT_NUM];
extern int numPoints;

main(argc,argv)
int argc;
char **argv;
{
    int i, j;
    int *num;
    char fname[64];
    FILE *fid;
    int val;

    if(argc != 2) {
        printf("usage: prepros file_name\n");
        exit(0);
    }

    readAll(argv[1]);
    init_graph();

    strcpy(fname, argv[1]);
    i=0;
    while( fname[i]!='.' && fname[i]!='\000' ) i++;
    fname[i] = '\0';
    strcat(fname, ".gph");
    printf("Try to write to %s\n", fname);
    fid = fopen(fname, "w");
    fprintf(fid, "%d ", numPoints);
    for( i=0; i<2; i++ )
        for( j=0; j<numPoints; j++ ) {
            fprintf(fid, "%d ", adjacent[i][j]);
        }
    fclose(fid);
    printf("Done\nBegin compare\n");
    fid = fopen(fname, "r");
    fscanf(fid, "%d ", &val);
    if( val!=numPoints ) printf("Error\n");
    for( i=0; i<2; i++ )
        for( j=0; j<numPoints; j++ ) {
            fscanf(fid, "%d ", &val);
            if( val != adjacent[i][j] ) printf("Error %d\n", i);
        }
    fclose(fid);
    printf("Compare done\n");
}

```

```

#include "radial_meas.h"
#include <stdio.h>

curve_dis radialDistances[MAX_CURVES];
int radialPnts[MAX_CURVES];
int source,numCurves;

void measRadial();
void pickRadialPnts();
void computeRadialPaths();
void displayRadialDistances();

void measRadial()
{
    int i;
    for (i=0;i<MAX_CURVES+1;i++)
        radialPnts[i] = -1;
    /* to differentiate between which is measured at the latest */
    numSegs = 0;
    pickSlc = -1;
    fprintf(stderr,"Inside measRadial(). *****\n");
    numCurves = 0;
    display();
    pickRadialPnts();
    computeRadialPaths();
    displayRadialDistances();
}

void pickRadialPnts()
{
    XEvent pickEvent;
    int i,done,curveNo;
    int first=1;
    point2 apt2;
    curveNo = 0;
    done = 0;
    while (!done)
    {
        XNextEvent(ncdisplay, &pickEvent);
        switch (pickEvent.type)
        {
            case ButtonPress :
                if (pickEvent.xbutton.window != ncwindow) break;
                if (pickEvent.xbutton.button == 3) {
                    numCurves = curveNo;
                    done = 1;
                    break;
                }
                if (pickEvent.xbutton.button == 2) {
                    if (curveNo == 0) break;
                    curveNo--;
                    XClearWindow(ncdisplay, ncwindow);
                    display();
                    highlight_point(source);
                }
            }
        }
    }
    for (i=0;i<curveNo;i++)
        highlight_point(radialPnts[i]);
}

if (pickEvent.xbutton.button == 1) {
    if (first) {
        source = pickPoint(pickEvent.xbutton.x,pi
ckEvent.xbutton.y);
        highlight_point(source);
        /*1
        projection(&(showPoints[source]), &apt2,
        XDrawArc(ncdisplay, ncwindow, ncgc, apt2
        /*
        first = 0;
        break;
    }
    radialPnts[curveNo] = pickPoint(pickEvent.xbutton
    highlight_point(radialPnts[curveNo]);
    curveNo++;
}
break;
default :
    break;
}
}
.x,pickEvent.xbutton.y);
}
computeRadialPaths()
{
    int i;
    for (i=0;i<numCurves;i++)
    {
        comp_path( source, radialPnts[i], &radialDistances[i] );
        highlight_curve(radialDistances[i]);
        fprintf(stderr,"length %d = %f\n", i,radialDistances[i].ds );
    }
    display();
}

void displayRadialDistances()
{
    int i;
    char length[40];
    for (i=0;i<numCurves;i++)
    {
        sprintf(length,"Curve %d = %6.2f inches.", i, lengthInches(radialDista
nces[i].ds) );
        XDrawString(ncdisplay, meas_win, meas_gc, 30, (i+1)*20, length, strlen(l
length) );
    }
}

```

```
#include <math.h>
#include <string.h>
#include "disp.h"

extern int adjacent[2][MAX_POINT_NUM];
extern int numPoints;

int readGraph(fname)
char *fname;
{
    int i, j;
    char gname[64];
    FILE *fid;
    int val;

    strcpy(gname, fname);
    i=0;
    while( gname[i]!='.' && gname[i]!='\000' ) i++;
    gname[i] = '\0';
    strcat(gname, ".gph");
    printf("Try to read %s\n", gname);
    fid = fopen(gname, "r");
    if( fid==NULL ) return(-1);
    fscanf(fid, "%d", &val);
    if( val!=numPoints ){
        printf("Error\nold numPoints=%d\nnew numPoints=%d\n", numPoints, val);
        return(-1);
    }
    for( i=0; i<2; i++ )
        for( j=0; j<numPoints; j++ )
            fscanf(fid, "%d", &adjacent[i][j]);
    return(1);
}
```

```

#include "region.h"

int slice_chest();
int slice_seat();
int slice_waist();
int s_moveup();
int s_movedown();
int s_max();
int s_min();
int centerfront();
int centerback();
int centerleft();
int centerleft();
int leftmost();
int rightmost();
int foremost();
int hindmost();
int p_moveup();
int p_movedown();
int p_clockwise();
int p_counter_clockwise();
int max_X();
int min_X();
int max_Z();
int min_Z();
double circumference();
double direct_dist();
double surface_dist();

int s_move();
int s_verify();

/*****
 * Region Operation Primitives
 *****/

region_all()
{
    int i;

    if( current_state.region != -1 ) free(cregion_des.slices);
    current_state.region = ALL;
    cregion_des.nsllices = numslices;
    cregion_des.slices = (int *) malloc(numslices*sizeof(int));
    for( i=0; i<numslices; i++ ) (cregion_des.slices)[i] = i;
}

region_arm(x)
int x;
{
    int i;
    int temp_set[200], temp_num;

    if( current_state.region != -1 ) free(cregion_des.slices);
    if( x==L ) current_state.region = LARM;
    else if( x==R ) current_state.region = RARM;
    else { printf("Incorrect arm region parameter.\n"); exit(0); }
    temp_num = 0;
    for( i=0; i<numslices; i++ )
        if( realSliceLen(i)<ARMCIRCUM )
            if( current_state.region==LARM ) {
                if( slice_center[i].y > 300.0 )
                    temp_set[temp_num++] = i;
            }
            else if( current_state.region==RLEG ) {
                if( slice_center[i].y <= 300.0 )
                    temp_set[temp_num++] = i;
            }
    cregion_des.nsllices = temp_num;
    cregion_des.slices = (int *) malloc(temp_num*sizeof(temp_num));
    for( i=0; i<temp_num; i++ ) (cregion_des.slices)[i] = temp_set[i];
}

region_torso()
{
    int i, done;
    int temp_set[200], temp_num;

    if( current_state.region != -1 ) free(cregion_des.slices);
    current_state.region = TORSO;
    temp_num = done = i = 0;
    while( done==0 && i<numslices-1 ) {
        if( fabs(slice_center[i].x-slice_center[i+1].x)>EPS ||
            fabs(slice_center[i].x-slice_center[i-1].x)>EPS )
            i++;
        else done = 1;
    }
    for( i<numslices; i++ )
        if( realSliceLen(i)>TORSOCIRCUM ) temp_set[temp_num++] = i;
    cregion_des.nsllices = temp_num;
    cregion_des.slices = (int *) malloc(temp_num*sizeof(temp_num));
    for( i=0; i<temp_num; i++ ) (cregion_des.slices)[i] = temp_set[i];
}

for general data files. */
if( slice_center[i].y > 300.0+175.0 )
    temp_set[temp_num++] = i;
}
else if( current_state.region==RARM ) {
    if( slice_center[i].y <= 300.0-150.0 )
        temp_set[temp_num++] = i;
}
}

cregion_des.nsllices = temp_num;
cregion_des.slices = (int *) malloc(temp_num*sizeof(temp_num));
for( i=0; i<temp_num; i++ ) (cregion_des.slices)[i] = temp_set[i];
}

region_leg(x)
int x;
{
    int i;
    double d;
    int temp_set[200], temp_num;

    if( current_state.region != -1 ) free(cregion_des.slices);
    region_torso();
    d = slice_center(cregion_des.slices(cregion_des.nsllices - 1)).x;
    free(cregion_des.slices);
    if( x==L ) current_state.region = LLEG;
    else if( x==R ) current_state.region = RLEG;
    else { printf("Incorrect leg region parameter.\n"); exit(0); }
    temp_num = 0;
    for( i=0; i<numslices; i++ )
        if( slice_center[i].x>d
            && realSliceLen(i)<LEGCIRCUM && realSliceLen(i)>ARMCIRCUM )
            if( current_state.region==LLEG ) {
                if( slice_center[i].y > 300.0 )
                    temp_set[temp_num++] = i;
            }
            else if( current_state.region==RLEG ) {
                if( slice_center[i].y <= 300.0 )
                    temp_set[temp_num++] = i;
            }
    cregion_des.nsllices = temp_num;
    cregion_des.slices = (int *) malloc(temp_num*sizeof(temp_num));
    for( i=0; i<temp_num; i++ ) (cregion_des.slices)[i] = temp_set[i];
}

region_torso()
{
    int i, done;
    int temp_set[200], temp_num;

    if( current_state.region != -1 ) free(cregion_des.slices);
    current_state.region = TORSO;
    temp_num = done = i = 0;
    while( done==0 && i<numslices-1 ) {
        if( fabs(slice_center[i].x-slice_center[i+1].x)>EPS ||
            fabs(slice_center[i].x-slice_center[i-1].x)>EPS )
            i++;
        else done = 1;
    }
    for( i<numslices; i++ )
        if( realSliceLen(i)>TORSOCIRCUM ) temp_set[temp_num++] = i;
    cregion_des.nsllices = temp_num;
    cregion_des.slices = (int *) malloc(temp_num*sizeof(temp_num));
    for( i=0; i<temp_num; i++ ) (cregion_des.slices)[i] = temp_set[i];
}

```

/* the following measurement standard is fine tuned
to fit mike.dat, a better measurement is desired

```

int s, p;
double x;
{
    return p_moveup(s, p, (-1.0)*x);
}

/* Move left x inches along surface */
/* Direction clockwise from top view */
int p_clockwise(s, p, x)
int s, p;
double x;
{
    int i, t;
    int symbol;
    double d, dl;

    if( x>0.0 ) symbol=1;
    else( symbol=0; x = x*(-1.0); )
    x = inchinlength(x);
    d = 0.0;
    i = p;
    while( d<x ){
        t = i;
        if( symbol ) i=(i+1-allSlices[s])%allSlices[s+1]-allSlices[s]
            +allSlices[s];
        else i=(i-1-allSlices[s])%allSlices[s+1]-allSlices[s]
            +allSlices[s];
        d += direct_dist(t, i);
    }
    return(t);
}

/* Move left x inches along surface */
/* Direction counter-clockwise from top view */
int p_counter_clockwise(s, p, x)
int s, p;
double x;
{
    return p_clockwise(s, p, (-1.0)*x);
}

int max_X(s, p1, p2)
int s, p1, p2;
{
    int i, t, num;
    double d, dl;

    if( p1<allSlices[s] || p2<allSlices[s]
        || p1>=allSlices[s+1] || p2>=allSlices[s+1] ){
        printf("max_X points not in slice\n");
        return(-1);
    }
    t = p1;
    i = t-1;
    num = allSlices[s+1]-allSlices[s];
    d = allPoints[t].y;
    do{
        i = (i+1-allSlices[s])%num + allSlices[s];
        dl = allPoints[i].y;
        if( dl>d ){ t=i; d=dl; }
    }while( i != p2 );
    return(t);
}

int min_X(s, p1, p2)
int s, p1, p2;
{
}

int max_Z(s, p1, p2)
int s, p1, p2;
{
}

int min_Z(s, p1, p2)
int s, p1, p2;
{
}

/*****
 * Measurement Primitives
 *****/
double circumference(s)
int s;
{
    return realSliceLen(s);
}

double direct_dist(p1, p2)
int p1, p2;
{
    /*display();
    showline(p1, p2);*/
    return distance(allPoints[p1],allPoints[p2]);
}

double surface_dist(p1, p2, cd)
int p1, p2;
curve_dis *cd;
{
    comp_path(p1, p2, cd);
    display();
    highlight_curve(cd);
    return cd->ds;
}

selectslice(percent)
double percent;
{
    int i;

    if( current_state.region == -1 ){
        printf("No current region available. Select a region first.\n");
        return;
    }
    if( 0.0>percent || 1.0<percent ){
        printf("Percentage out of bound.\n");
        return;
    }
    i = (int)(percent * (double)(cregion_des.nslices-1));
    current_state.slice = cregion_des.slices[i];
    showslice(current_state.slice);
}

```

```

(
    return s_move(&region_des, s, (-1.0)*x);
)

/* Return entry to the slice array */
int s_movedown(s, x)
int s;
double x;
{
    return s_move(&region_des, s, x);
}

/* Verify that slices x1 and x2 are in region r */
int s_verify(r, x1, x2, idx1, idx2)
int x1, x2;
region *r;
int *idx1, *idx2;
{
    int i, t, t1, t2;

    if( x1<0 || x2<0 || x1>=numSlices || x2>=numSlices ){
        printf("Parameters to s_max domain out of bound\n");
        return(-1);
    }
    i = 0;
    t1 = -1;
    while( t1== -1 && i<r->nslices ){
        if( r->sllices[i]==x1 ) t1=i;
        i++;
    }
    if( t1== -1 ){ printf("s_max: x1 not in region r.\n"); return(-1); }
    i = 0;
    t2 = -1;
    while( t2== -1 && i<r->nslices ){
        if( r->sllices[i]==x2 ) t2=i;
        i++;
    }
    if( t2== -1 ){ printf("s_max: x2 not in region r.\n"); return(-1); }
    *idx1 = t1; *idx2 = t2;
    printf("entries are %d and %d\n", t1, t2);
    return(0);
}

/* Find the slice with the max circumference inside region r,
and between slices x1 and x2. */
int s_max(x1, x2)
int x1, x2;
{
    double d, dl;
    int t, i, j, t1, t2;
    region *r;

    r = &region_des;
    if( s_verify(r, x1, x2, &t1, &t2) ) return(-1);
    if( x1==x2 ) return(x1);

    d = realSliceLen(x1);
    t = x1;
    i = t1;
    while( i<t2 ){
        i++;
        j = r->sllices[i];
        d = fabs( allPoints[t].y - slice_center[s].y );
        for( i=t+1; i<allSlices[i]; i++ ){
            dl = fabs( allPoints[i].y - slice_center[s].y );
            if( dl<d && (allPoints[i].z - slice_center[s].z)>0.0 ){
                t=i; d=dl;
            }
        }
        return(t);
    }

    int centerfront(s)
    int s;
    {
        int i, t;
        double d, dl;

        t = allSlices[s];
        d = fabs( allPoints[t].y - slice_center[s].y );
        for( i=t+1; i<allSlices[i]; i++ ){
            dl = fabs( allPoints[i].y - slice_center[s].y );
            if( dl<d && (allPoints[i].z - slice_center[s].z)>0.0 ){
                t=i; d=dl;
            }
        }
        return(t);
    }

    int centerback(s)
    int s;
    {
        int i, t;
        double d, dl;

        t = allSlices[s];
        d = fabs( allPoints[t].y - slice_center[s].y );
        for( i=t+1; i<allSlices[i]; i++ ){
            dl = fabs( allPoints[i].y - slice_center[s].y );
            if( dl<d && (allPoints[i].z - slice_center[s].z)>0.0 ){
                t=i; d=dl;
            }
        }
        return(t);
    }
}

```

```

*****
int slice_chest()
{
    int i, notdone;

    notdone = i = 1;
    while( notdone && i<numslices-1 ){
        if( fabs(slice_center[i].x-slice_center[i+1].x)<EPS &&
            fabs(slice_center[i].x-slice_center[i+1].x)<EPS ){
            return(i);
        }
        else i++;
    }
    return(-1);
}

int slice_seat()
{
}

int slice_waist()
{
}

/* Move x inches from slice s inside region r */
/* Return entry to the slice array */
int s_move(r, s, x)
int s;
double x;
region *r;
{
    double xdest, d, d1;
    int i, j, k, temp[10], t, count;

    xdest = slice_center[s].x + inchInLength(x);
    t = r->slices[0];
    for( k=1; k<r->nslices; k++ ){
        i = r->slices[k];
        if( fabs(slice_center[i].x-xdest)<fabs(slice_center[t].x-xdest) )
            t=i;
    }
    count=0;
    for( k=1; k<r->nslices; k++ ){
        i = r->slices[k];
        if( fabs(slice_center[i].x-slice_center[t].x)<EPS )
            temp[count++]=i;
    }
    d = sqrt((slice_center[t].y-slice_center[s].y)*(slice_center[t].y-slice_center[s].y) +
        (slice_center[t].z-slice_center[s].z)*(slice_center[t].z-slice_center[s].z));
    for(i=0; i<count; i++){
        j = temp[i];
        d1 = sqrt((slice_center[j].y-slice_center[s].y)*(slice_center[j].y-slice_center[s].y) +
            (slice_center[j].z-slice_center[s].z)*(slice_center[j].z-slice_center[s].z));
        if( d1<d ) t=j;
    }
    return(t);
}

/* Return entry to the slice array */
int s_moveup(s, x)
int s;
double x;

```



```

    if( dl<d && (allPoints[i].z - slice_center[s].z)<0.0 ) {
        t=i; d=dl;
    }
    return(t);
}

int centerleft(s)
int s;
{
    int i, t;
    double d, dl;

    t = allSlices[s];
    d = fabs( allPoints[t].z - slice_center[s].z );
    for( i=t+1; i<allSlices[s+1]; i++ ){
        dl = fabs( allPoints[i].z - slice_center[s].z );
        if( dl<d && (allPoints[i].y - slice_center[s].y)>0.0 ) {
            t=i; d=dl;
        }
    }
    return(t);
}

int centerright(s)
int s;
{
    int i, t;
    double d, dl;

    t = allSlices[s];
    d = fabs( allPoints[t].z - slice_center[s].z );
    for( i=t+1; i<allSlices[s+1]; i++ ){
        dl = fabs( allPoints[i].z - slice_center[s].z );
        if( dl<d && (allPoints[i].y - slice_center[s].y)<0.0 ) {
            t=i; d=dl;
        }
    }
    return(t);
}

int rightmost(s)
int s;
{
    int i, t;
    double d, dl;

    t = allSlices[s];
    d = allPoints[t].y - slice_center[s].y;
    for( i=t+1; i<allSlices[s+1]; i++ ){
        dl = allPoints[i].y - slice_center[s].y;
        if( dl<d ) {
            t=i; d=dl;
        }
    }
    return(t);
}

int leftmost(s)
int s;
{
    int i, t;
    double d, dl;

    ss = s_moveup(s, x);
    t = allSlices[ss];
    d = yz_dist(t, p);
    for( i=t+1; i<allSlices[ss+1]; i++ ){
        dl = yz_dist(i, p);
        if( dl<d ) { t=i; d=dl; }
    }
    current_state.slice = ss;
    return(t);
}

int p_movedown(s, p, x)

```

```

#include "slice_menu.h"

char *views[] = { "Vertical Y", "Vertical Z", "Collapsed View" };
Window sliceMenu_win, slice_wins[ MAX_VIEWS+1 ];
GC sliceMenu_gc, slice_gcs[ MAX_VIEWS+1 ];
XColor fg_col, bg_col;
int sliceCnt;
int cur_slice=5;

Initialize_sliceMenu()
{
    XFontStruct *fontStr;
    int i;

    sliceCnt = MAX_VIEWS;
    create_slice_menu_windows(&sliceMenu_win, &sliceMenu_gc, slice_wins, slice_gcs, sliceCnt);

    fontStr = XLoadQueryFont( ncdisplay, "-adobe-courier-bold-r-normal--0-0-75-75-m-0-iso8859-1" );
    if (fontStr != 0)
        for (i=0; i<sliceCnt; i++)
            XSetFont( ncdisplay, slice_gcs[i], fontStr->fid );
}

refreshsliceMenu()
{
    int i;

    for (i=0; i<sliceCnt; i++)
        XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
    XDrawString(ncdisplay, slice_wins[sliceCnt], slice_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("Done"));
}

mapsliceMenu(x,y)
int x,y;
{
    int i;
    XEvent df_event;
    XWindowAttributes winAttrs;

    XMapRaised(ncdisplay, sliceMenu_win);
    for (i=0; i<sliceCnt+1; i++)
        XMapRaised(ncdisplay, slice_wins[i]);

    XGetWindowAttributes( ncdisplay, slice_wins[sliceCnt], &winAttrs );
    if (winAttrs.map_state != 2)
        XWindowEvent( ncdisplay, slice_wins[sliceCnt], ExposureMask, &df_event );

    redrawsliceMenu();
}

redrawsliceMenu()
{
    int i;

    for (i=0; i<sliceCnt; i++)
        XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
}

",strlen("Done"));
XFlush( ncdisplay );
}

sliceMenu()
{
    int i;
    char filename[MAX_FILE_NAME];

    switch (nccvent.type)
    {
        case Expose :
            for (i=0; i<sliceCnt+1; i++)
                if (nccvent.xexpose.window == slice_wins[i]) break;
            if ( i < sliceCnt+1 )
                for (i=0; i<sliceCnt; i++)
                    XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
            XDrawString(ncdisplay, slice_wins[sliceCnt], slice_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("Done"));
            break;

        case MapNotify :
            if (nccvent.xmap.window == sliceMenu_win)
                for (i=0; i<sliceCnt; i++)
                    XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
            XDrawString(ncdisplay, slice_wins[sliceCnt], slice_gcs[0], STR_LOC_X, STR_LOC_Y, "Done", strlen("None"));
            break;

        case ButtonPress :
            for (i=0; i<sliceCnt; i++)
                if (nccvent.xbutton.subwindow == slice_wins[i]) break;
            if ( i == sliceCnt )
                XUnmapWindow( ncdisplay, sliceMenu_win );
                XFlush( ncdisplay );
            else
                XSetForeground( ncdisplay, slice_gcs[i], bg_col.pixel );
                XSetWindowBackground( ncdisplay, slice_wins[i], fg_col.pixel );
                XClearWindow( ncdisplay, slice_wins[i] );
                XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
                XFlush( ncdisplay );

            manage_slice_menu(i);

            XSetForeground( ncdisplay, slice_gcs[i], fg_col.pixel );
            XSetWindowBackground( ncdisplay, slice_wins[i], bg_col.pixel );
            XClearWindow( ncdisplay, slice_wins[i] );
            XDrawString(ncdisplay, slice_wins[i], slice_gcs[i], STR_LOC_X, STR_LOC_Y, views[i], strlen(views[i]));
            XFlush( ncdisplay );
    }
}

```

```

        slice_hints[i].height, 0, depth, InputOutput, CopyFromParent, slice_win_mask
        , &slice_win_attrs);
    }

    /* Create a graphics context */
    *gc = XCreateGC ( ncdisplay, *win, 0, 0);
    XSetBackground (ncdisplay, *gc, bg_col.pixel);
    XSetWindowBackground (ncdisplay, *win, bg_col.pixel);
    XSetForeground (ncdisplay, *gc, fg_col.pixel);
    XSelectInput (ncdisplay, *win, ButtonPressMask | ExposureMask | StructureNotifyMask);

    for (i=0; i<num_files+1; i++) {
        slice_gcs[i] = XCreateGC ( ncdisplay, slice_wins[i], 0, 0);
        XSetBackground (ncdisplay, slice_gcs[i], bg_col.pixel);
        XSetWindowBackground (ncdisplay, slice_wins[i], bg_col.pixel);
        XSetForeground (ncdisplay, slice_gcs[i], fg_col.pixel);
        XSelectInput (ncdisplay, slice_wins[i], ExposureMask | StructureNotifyMask);
    }

    sk);

    manage_slice_menu(item_no)
    int item_no;
    {
        switch (item_no)
        {
            case 0:    display_vertical_slice(Y_AXIS);
                      break;

            case 1:    display_vertical_slice(Z_AXIS);
                      break;

            case 2:    display_collapsed_slices();
                      break;

            default:    break;
        }
    }

create_slice_menu_windows(win, gc, slice_wins, slice_gcs, num_files)
Window *win, *slice_wins;
GC *gc, *slice_gcs;
int num_files;
{
    XSetWindowAttributes slice_win_attrs;
    unsigned long slice_win_mask;
    XSizeHints slice_hint, slice_hints[MAX_VIEWS];
    int i, depth;

    depth = XDisplayPlanes( ncdisplay, ncscreen);

    bg_col.red    = slice_BG_RED;
    bg_col.blue   = slice_BG_BLUE;
    bg_col.green   = slice_BG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &bg_col);

    fg_col.red    = slice_FG_RED;
    fg_col.blue   = slice_FG_BLUE;
    fg_col.green   = slice_FG_GREEN;
    XAllocColor(ncdisplay, nc_cmap, &fg_col);

    slice_win_attrs.border_pixel = fg_col.pixel;
    slice_win_attrs.background_pixel = bg_col.pixel;
    slice_win_attrs.override_redirect = True;

    slice_win_mask = ( CWBorderPixel | CWOverrideRedirect );

    /* set the window parameters for slice window */
    slice_hint.x = 800; slice_hint.y = 450;
    slice_hint.width = 120; slice_hint.height = (num_files+1) * FN_WIDTH;
    slice_hint.flags = USPosition | PSize;

    /* Create the window with the above given specifications */
    *win = XCreateSimpleWindow ( ncdisplay, DefaultRootWindow ( ncdisplay ), slice_h
int.x, slice_hint.y,
                                slice_hint.width, slice_hint.height, 5, fg_col.pixel, bg
_col.pixel);
    XSetStandardProperties( ncdisplay, *win, "SLICE VIEWS", "SLICE VIEWS", None, 0,
0, &slice_hint );

    /* set the window parameters for slice sub windows */
    for (i=0; i<num_files+1; i++) {
        slice_hints[i].x = 0;
        slice_hints[i].y = i*FN_WIDTH;
        slice_hints[i].width = 120; slice_hints[i].height = FN_WIDTH;
        slice_hints[i].flags = USPosition | PSize;

        slice_wins[i] = XCreateWindow(ncdisplay, *win, slice_hints[i].x, slice_hints[i].y,
slice_hints[i].width,

```

```

#include "disp.h"
#include "slice_view.h"

#define EPS0 5.0
#define EPS1 12.0
#define EPS2 30.0

/*****
 * This structure is used in
 * dijkstra.c, it returns the
 * slice number which a point
 * belongs to.
 *****/
extern int inslice[MAX_POINT_NUM];

int in_range();

/*****
 * Highlight slices vertically.
 *****/
slice_view_vertical(point, h_points, axis)
int point, axis;
disp_pts *h_points;
{
    int i, j;
    int num, slice, count, select1, select2;

    num = slice = 0;
    for( i=0; i<numSlices; i++ ){
        count = in_range(point, i, axis, &select1, &select2);
        if( count > 0 ) dpoints[num++] = select1;
        if( count == 2 ) dpoints[num++] = select2;
    }
    h_points->num = num;
    h_points->points = (int *) malloc(num*sizeof(i));
    for( i=0; i<num; i++ )
        h_points->points[i] = dpoints[i];
}

int in_range(point, slice, axis, select1, select2)
int point, slice, *select1, *select2, axis;
{
    int i, j;
    int bestp;
    double bestdist, d;
    int first, last;

    /* Select the 1st point */
    first = allSlices[slice];
    last = allSlices[slice+1];
    bestdist = 100000.0; bestp = first;
    for( i=first; i<last; i++ )
        if( axis==Y_AXIS )
            for( i=first; i<last; i++ ){
                d = fabs(allPoints[i].y - allPoints[point].y);
                if( d<bestdist && d<EPS0 ){
                    bestdist = d;
                    bestp = i;
                }
            }
    else if( axis==Z_AXIS )
        for( i=first; i<last; i++ ){
            d = fabs(allPoints[i].z - allPoints[point].z);
            if( d<bestdist && d<EPS2 ){
                bestdist = d;
                bestp = i;
            }
        }
    if( d<bestdist && d<EPS2 ){
        bestdist = d;
        bestp = i;
    }
}

if( bestdist>99999.0 ) return(0);
*select1 = bestp;
/* Now select the 2nd point */
bestdist = 100000.0; bestp = first;
if( axis==Y_AXIS )
    for( i=first; i<last; i++ ){
        d = fabs(allPoints[i].y - allPoints[point].y);
        if( i!=*select1
            && fabs(allPoints[*select1].z - allPoints[i].z)>EPS1
            && d<bestdist && d<EPS0*2.0 ){
            bestdist = d;
            bestp = i;
        }
    }
else if( axis==Z_AXIS )
    for( i=first; i<last; i++ ){
        d = fabs(allPoints[i].z - allPoints[point].z);
        if( i!=*select1
            && fabs(allPoints[*select1].x - allPoints[i].x)>EPS1
            && d<bestdist && d<EPS2*2.0 ){
            bestdist = d;
            bestp = i;
        }
    }
if( bestdist>99999.0 ) return(1);
*select2 = bestp;
return(2);
}

/*****
 * Collapse several slices together.
 * Implementation details are to
 * be changed later.
 *****/
collapse(colpSlices, h_points)
collapse_slices *colpSlices;
disp_pts *h_points;
{
    int j, k;
    int slice, first, last;
    int dpoints[1500];
    int num;

    num = 0;
    for( k=0; k<colpSlices->num; k++ ){
        slice = colpSlices->slices[k];
        first = allSlices[slice];
        last = allSlices[slice+1];
        for( j=first; j<last; j++ ) dpoints[num++] = j;
    }
    colpSlices->circumference = 100.0;
    h_points->num = num;
    h_points->points = (int *) malloc(num*sizeof(k));
    for( k=0; k<num; k++ ) h_points->points[k] = dpoints[k];
}

```

```

#include <X11/Xatom.h>
#include <X11/X.h>
#include "startx.h"

/* Initialize the name of the Window */
char winname[] = "DLA 3-D NONCONTACT MEASUREMENT PROJECT ";

/* Declarations of variables required to display on an X-Window */

Display *ncdisplay;
Window ncwindow, menu_window, meas_win, cur_fname_win;
Window view_win;
GC ncgc, menu_gc, menu_xor_gc, menu_pm_gc, meas_gc, cur_fname_gc, view_gc;
XSizeHints nchint, win_hint;
int ncscreen;
int depth;
unsigned long hfg, hbg;

/* C O L O R */
Colormap nc_cmap;

XPoint scr_coords[MAX_POINT_NUM];

double chestLen, waistLen, seatLen;

/* The X-Window initializations such as creating a window, setting the foreground
/* and background colors, selection of the input events etc. is done in this routine */

Initialize_X()
{
    Status result;
    XColor fg_col, bg_col;

    /* open the default Display */
    ncdisplay = XOpenDisplay ("");

    ncscreen = DefaultScreen ( ncdisplay );
    depth = XDisplayPlanes( ncdisplay, ncscreen);

    nc_cmap = DefaultColormap( ncdisplay, ncscreen );

    /* set the window parameters */
    nchint.x = 50; nchint.y = 100;
    nchint.width = WIN_WIDTH; nchint.height = WIN_HEIGHT;
    nchint.flags = USPosition | PSize;

    /* Create the window with the above given specifications */
    ncwindow = XCreateSimpleWindow ( ncdisplay,
                                     DefaultRootWindow ( ncdisplay ), nchint.x, nchint.y,
                                     nchint.width, nchint.height, 5, hfg, hbg);

    /* Set the standard properties of the window */
    XSetStandardProperties( ncdisplay, ncwindow, winname, winname,
                           None, 0, 0, &nchint );

    /* Create a graphics context */
    ncgc = XCreateGC ( ncdisplay, ncwindow, 0, 0);
    XSetLineAttributes(ncdisplay, ncgc, 2, LineSolid, 0, JoinRound );
}

/* Select all the input events to be processed */
XSelectInput (ncdisplay, ncwindow,
              ButtonReleaseMask | ButtonPressMask | KeymapStateMask | KeyPressMask | ExposureMask );

ask );

fg_col.red = WIN_FG_RED;
fg_col.blue = WIN_FG_BLUE;
fg_col.green = WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Foreground color not allocated.\n");
XSetForeground (ncdisplay, ncgc, fg_col.pixel);

bg_col.red = WIN_BG_RED;
bg_col.blue = WIN_BG_BLUE;
bg_col.green = WIN_BG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &bg_col);
if (!result) {
    printf("Background color not allocated.\n");
    exit(0);
}
XSetBackground (ncdisplay, ncgc, bg_col.pixel);
XSetWindowBackground (ncdisplay, ncwindow, bg_col.pixel);
}

/* The X-Window initializations such as creating a window, setting the foreground
/* and background colors, selection of the input events etc. is done in this routine */

Initialize_windows()
{
    unsigned long win_mask;
    int imageWinWidth, imageWinHeight;
    unsigned long valueMask;
    XGCValues values;
    XFontStruct *fontStr;
    XColor col, fg_col, bg_col;
    Status result;

    /* set the window parameters for current filename window */
    win_hint.x = 50 + WIN_WIDTH - 207;
    win_hint.y = 103 + WIN_HEIGHT - 60;
    win_hint.width = CUR_FNAME_WIN_WIDTH; win_hint.height = CUR_FNAME_WIN_HEIGHT;
    win_hint.flags = USPosition | PSize;

    /* Create the window with the above given specifications */
    cur_fname_win = XCreateSimpleWindow( ncdisplay, DefaultRootWindow(ncdisplay), win
                                     _hint.x,
                                     win_hint.y, win_hint.width, win_hint.height, 2, hfg, hbg);

    strcpy(winname, "IMAGE NAME");

    /* Set the standard properties of the window */
    XSetStandardProperties( ncdisplay, cur_fname_win, winname, winname,
                           None, 0, 0, &win_hint );

    cur_fname_gc = XCreateGC ( ncdisplay, cur_fname_win, 0, 0);
    fontStr = XLoadQueryFont ( ncdisplay, "rk16");
    if (fontStr != 0)
        XSetFont (ncdisplay, cur_fname_gc, fontStr->fid);
}

```

```

fg_col.red = MENU_WIN_FG_RED;
fg_col.blue = MENU_WIN_FG_BLUE;
fg_col.green = MENU_WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Foreground color not allocated.\n");
XSetForeground( ncdisplay, cur_fname_gc, fg_col.pixel);

bg_col.red = MENU_WIN_BG_RED;
bg_col.blue = MENU_WIN_BG_BLUE;
bg_col.green = MENU_WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &bg_col);
if (!result)
    printf("Background color not allocated.\n");
XSetBackground( ncdisplay, cur_fname_gc, bg_col.pixel);

/* set the window parameters for menu window */
win_hint.x = 50 + WIN_WIDTH + 25;
win_hint.y = 3;
/*win_hint.y = 101 + MEAS_WIN_HEIGHT + 40;*/
win_hint.width = MENU_WIN_WIDTH; win_hint.height = MENU_WIN_HEIGHT;
win_hint.flags = USPosition | PSize;

/* Create the window with the above given specifications */
menu_window = XCreateSimpleWindow( ncdisplay, DefaultRootWindow(ncdisplay), win_
    _hint.x, win_hint.y, win_hint.width, win_hint.height, 2, hfg, hbg);

strcpy(winname, "MENU");

/* Set the standard properties of the window */
XSetStandardProperties( ncdisplay, menu_window, winname, winname,
    None, 0, 0, &win_hint );

/* Create a graphics context */
menu_gc = XCreateGC( ncdisplay, menu_window, 0, 0);
menu_pm_gc = XCreateGC( ncdisplay, menu_window, 0, 0);

XSetLineAttributes(ncdisplay, menu_gc, 2, LineSolid, 0, JoinRound);
XSetLineAttributes(ncdisplay, menu_pm_gc, 2, LineSolid, 0, JoinRound);

fontStr = XLoadQueryFont( ncdisplay, "-schumacher-clean-bold-r-normal--14-140-75-
75-c-80-iso8859-1");
if (fontStr != 0)
    XSetFont( ncdisplay, menu_gc, fontStr->fid);

fontStr = XLoadQueryFont( ncdisplay, "lucidasanspewriter-bold-10");
if (fontStr != 0)
    XSetFont( ncdisplay, menu_pm_gc, fontStr->fid);

fg_col.red = MENU_WIN_FG_RED;
fg_col.blue = MENU_WIN_FG_BLUE;
fg_col.green = MENU_WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Foreground color not allocated.\n");
XSetForeground( ncdisplay, menu_gc, fg_col.pixel);
XSetBackground( ncdisplay, menu_pm_gc, fg_col.pixel);

bg_col.red = MENU_WIN_BG_RED;
bg_col.blue = MENU_WIN_BG_BLUE;
bg_col.green = MENU_WIN_BG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &bg_col);
if (!result)
    printf("Background color not allocated.\n");
XSetBackground( ncdisplay, meas_gc, fg_col.pixel);
XSetWindowBackground( ncdisplay, meas_win, bg_col.pixel);

fg_col.red = MENU_WIN_FG_RED;
fg_col.blue = MENU_WIN_BG_BLUE;
fg_col.green = MENU_WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Background color not allocated.\n");
exit(0);

XsetBackground( ncdisplay, menu_gc, bg_col.pixel);
XsetBackground( ncdisplay, menu_pm_gc, bg_col.pixel);
XsetWindowBackground( ncdisplay, menu_window, bg_col.pixel);

/* settings for XOR gc for the menu window */
valueMask = GCFUNCTION;
values.foreground = fg_col.pixel;
values.function = GXxor;
menu_xor_gc = XCreateGC( ncdisplay, menu_window, valueMask, &values );
XsetForeground( ncdisplay, menu_xor_gc, fg_col.pixel);
XsetBackground( ncdisplay, menu_xor_gc, bg_col.pixel);

/* set the window parameters for measurements window */
win_hint.x = 50 + WIN_WIDTH;
win_hint.y = 30;
win_hint.width = MEAS_WIN_WIDTH; win_hint.height = MEAS_WIN_HEIGHT;
win_hint.flags = USPosition | PSize;

/* Create the window with the above given specifications */
meas_win = XCreateSimpleWindow( ncdisplay, DefaultRootWindow(ncdisplay), win_
    _hint.x, win_hint.y, win_hint.width, win_hint.height, 2, hfg, hbg);

strcpy(winname, "MEASUREMENTS");

/* Set the standard properties of the window */
XSetStandardProperties( ncdisplay, meas_win, winname, winname,
    None, 0, 0, &win_hint );

meas_gc = XCreateGC( ncdisplay, meas_win, 0, 0);
fontStr = XLoadQueryFont( ncdisplay, "-schumacher-clean-bold-r-normal--16-160-75-
5-c-80-iso8859-1");
if (fontStr != 0)
    XSetFont( ncdisplay, meas_gc, fontStr->fid);

fg_col.red = MENU_WIN_FG_RED;
fg_col.blue = MENU_WIN_FG_BLUE;
fg_col.green = MENU_WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Foreground color not allocated.\n");
XsetForeground( ncdisplay, meas_gc, fg_col.pixel);

bg_col.red = MENU_WIN_BG_RED;
bg_col.blue = MENU_WIN_BG_BLUE;
bg_col.green = MENU_WIN_BG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &bg_col);
if (!result)
    printf("Background color not allocated.\n");
exit(0);

XsetBackground( ncdisplay, meas_gc, bg_col.pixel);
XsetWindowBackground( ncdisplay, meas_win, bg_col.pixel);

```

```

/* set the window parameters for view window */
win_hint.x = 50 + WIN_WIDTH + 25;
win_hint.y = 41 + MENU_WIN_HEIGHT;
win_hint.width = MENU_WIN_WIDTH; win_hint.height = MENU_WIN_HEIGHT;
win_hint.flags = USposition | PSize;

/* Create the window with the above given specifications */
view_win = XCreateSimpleWindow ( ncdisplay, DefaultRootWindow(ncdisplay), win_hi
nt.x, win_hint.y,
    win_hint.width, win_hint.height, 2, hfg, hbg);

strcpy(winname,"VIEWS");

/* Set the standard properties of the window */
XSetStandardProperties( ncdisplay, view_win, winname, winname,
    None, 0, 0, &win_hint );

/* Create a graphics context */
view_gc = XCreateGC ( ncdisplay, view_win, 0, 0);

fontStr = XLoadQueryFont ( ncdisplay, "-schumacher-clean-bold-r-normal--14-140-75-
75-c-80-iso8859-1");
if (fontStr != 0)
    XSetFont (ncdisplay, view_gc, fontStr->fid);

fg_col.red = WIN_FG_RED;
fg_col.blue = WIN_FG_BLUE;
fg_col.green = WIN_FG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &fg_col);
if (!result)
    printf("Foreground color not allocated.\n");
XSetForeground (ncdisplay, view_gc, fg_col.pixel);

bg_col.red = WIN_BG_RED;
bg_col.blue = WIN_BG_BLUE;
bg_col.green = WIN_BG_GREEN;
result = XAllocColor( ncdisplay, nc_cmap, &bg_col);
if (!result) {
    printf("Background color not allocated.\n");
    exit(0);
}
XSetBackground (ncdisplay, view_gc, bg_col.pixel);
XSetWindowBackground (ncdisplay, view_win, bg_col.pixel);

butCol1.red = BUT_RED1;
butCol1.blue = BUT_BLUE1;
butCol1.green = BUT_GREEN1;
result = XAllocColor( ncdisplay, nc_cmap, &butCol1);

butCol2.red = BUT_RED2;
butCol2.blue = BUT_BLUE2;
butCol2.green = BUT_GREEN2;
result = XAllocColor( ncdisplay, nc_cmap, &butCol2);

butCol3.red = BUT_RED3;
butCol3.blue = BUT_BLUE3;
butCol3.green = BUT_GREEN3;
result = XAllocColor( ncdisplay, nc_cmap, &butCol3);

/* Select all the input events to be processed */
XSelectInput (ncdisplay, menu_window,
    ButtonPressMask | ButtonReleaseMask | ExposureMask |
    ButtonMotionMask | EnterWindowMask );
XSelectInput (ncdisplay, meas_win, ExposureMask );
XSelectInput (ncdisplay, cur_fname_win, ExposureMask );
XSelectInput (ncdisplay, view_win, ExposureMask );

XMapRaised (ncdisplay, ncwindow);
XMapRaised (ncdisplay, menu_window);
XMapRaised (ncdisplay, meas_win);
XMapRaised (ncdisplay, cur_fname_win);
XMapRaised (ncdisplay, view_win);

int i, regno;
int xpos, ypos;
point2 apt2;
int c, s, p;
double inch;

XClearWindow( ncdisplay, ncwindow );

for( i=0; i<numPoints; i++) {
    projection(&showPoints[i], &apt2, i);
    scr_coords[i].x = apt2.x;
    scr_coords[i].y = apt2.y;
};
XDrawPoints(ncdisplay,ncwindow,ncgc,scr_coords,numPoints,CoordModeOrigin);

do{
    printf("Choose a region\n");
    do{
        printf("1=1Arm, 2=rArm, 3=lLeg, 4=rLeg, 5=torso, 6=upBody\n");
        printf("9=Slice operations, 0=Quit\n");
        scanf("%d", &regno);
        if( regno==0 ) return;
        if( regno!=9 ){ display(); dispRegion(regno); XFlush(ncdisplay); }
    }while( regno!=9 );
    printf("Now slice operations.\n");
    s = cregion_des.slices[0];
    display(); highlight(s); XFlush(ncdisplay);
    do{
        printf("Choose a slice.\n");
        do{
            printf("1=move up, 2=move down, 3=point selection, 4=End, 0=Quit\n");
            scanf("%d", &c);
            if( !c ) return;
            if( c==2 && c>0 ){
                printf("distance(inch)=");
                scanf("%lf", &inch);
                if( c==1 ) s=s_moveup(s, inch);
                else s = s_movedown(s, inch);
            }
            display(); highlight(s); XFlush(ncdisplay);
        }while( !(c==4 || c==3) );
        if( c==4 ) continue;
    }do{
        printf("Now point operations.");
        printf("1=cf, 2=cb, 3=cl, 4=cr, 5=fm, 6=hm, 7=lm, 8=rm, 9=End, 0=Quit\n");
        scanf("%d", &c);
    }
}

```

```

if( !c ) return;
if( c==9 ) continue;
switch( c ){
    case 1 : p=centerfront(s); break;
    case 2 : p=centerback(s); break;
    case 3 : p=centerleft(s); break;
    case 4 : p=centerright(s); break;
    case 5 : p=foremost(s); break;
    case 6 : p=hindmost(s); break;
    case 7 : p=leftmost(s); break;
    case 8 : p=rightmost(s); break;
}
display(); highlight_point(p); XFlush(ncdisplay);
do(
    printf("point move: 1=up, 2=down, 3=clockwise, 4=counter-c-wise, 5=End
, 0=Quit\n");
    scanf("%d", &c);
    if( !c ) return;
    if( c==5 ) continue;
    printf("distance(inch) = ");
    scanf("%lf", &inch);
    switch( c ){
        case 1 : p=p_moveup(s, p, inch); break;
        case 2 : p=p_movedown(s, p, inch); break;
        case 3 : p=p_clockwise(s, p, inch); break;
        case 4 : p=p_counter_clockwise(s, p, inch); break;
    }
    display(); highlight_point(p); XFlush(ncdisplay);
    while( c!=5 );
    while( c!=9 );
    while( c!=4 );
    XFlush(ncdisplay);
}
*/
showregion(regno)
int regno;
{
    display(); dispRegion(regno); XFlush(ncdisplay);
}
showslice(s)
int s;
{
    display(); highlight(s); XFlush(ncdisplay);
}
showpoint(p)
int p;
{
    display(); highlight_point(p); XFlush(ncdisplay);
}
showline(p1, p2)
int p1, p2;
{
    display(); highlight_line(p1, p2); /*XFlush(ncdisplay);*/
}
/*****
This routine displays the points to the screen.
*****/
display()
{
    int i;
    int xpos, ypos;
    point2 apt2;
    XClearWindow(ncdisplay, ncwindow);
    for( i=0; i<numPoints; i++) {
        projection(&(showPoints[i]), &apt2, i);
        scr_coords[i].x = apt2.x;
        scr_coords[i].y = apt2.y;
    };
    XDrawPoints(ncdisplay, ncwindow, ncgc, scr_coords, numPoints, CoordModeOrigin);
    highlightMeasmts();
    XClearWindow(ncdisplay, meas_win);
    dispMeasmts();
    if( thefirst ){ dispCenters(); centers(allPoints); thefirst=0; }
    /*dispRegion();*/
    if( is_vert_slice )
        show_vert_slice(&vert_slice, cur_axis);
    if( is_colp_slice )
        showCollpSlices(&collaps_slice);
    XFlush(ncdisplay);
}
highlightMeasmts()
{
    int i;
    if( pickSlc!=-1 ) highlight(pickSlc);
    if( pickPnt[0]!=-1 ) highlight_point(pickPnt[0]);
    if( pickPnt[1]!=-1 ) highlight_point(pickPnt[1]);
    if( pickPnt[0]!=-1 && pickPnt[1]!=-1 ) {
        /* highlight_line(pickPnt[0], pickPnt[1]); */
    }
    if( numSegs > 0 )
        for( i=0; i<numSegs; i++)
            highlight_curve( &cDistances[i] );
    if( numCurves > 0 )
        for( i=0; i<numCurves; i++)
            highlight_curve( &radialDistances[i] );
}
calculate_measmts()
{
    chestLen = sliceLen(chest);
    waistLen = sliceLen(waist);
    seatLen = sliceLen(seat);
}
dispMeasmts()
{
    char msg[80];
    if( pickSlc!=-1 ) {
        sprintf(msg, " SLICE LENGTH = %.2f inches", sliceLen(pickSlc));
        XDrawString(ncdisplay, meas_win, meas_gc, 25, 160, msg, strlen(msg));
        printf("Center at %f %f", circum=&f\n", slice_center[pickSlc].x, slice_

```


startx.c Sat Dec 4 15:37:07 1993 5

```

center(pickslc].y, slice_center[pickslc].z, realSliceLen(pickslc));
    }
    else
        if( pickPnt[0]==-1 && pickPnt[1]==-1 ) {
            sprintf(msg," DIRECT DISTANCE = %6.2f inches", pointLen(pickPnt[0], pi
ckPnt[1]));
            XDrawString( ncdisplay, meas_win, meas_gc, 25, 160, msg, strlen(msg));
        }
    else
        if (numSegs > 0) {
            printf("total_distance = %6.2f inches\n", lengthInInches(total_seq_len)
);
            printf(msg," TOTAL DISTANCE = %6.2f inches", lengthInInches(total_seq
_len));
            XDrawString( ncdisplay, meas_win, meas_gc, 25, 160, msg, strlen(msg));
        }
    }

dispCenters()
{
    int i, j;
    int x1, y1, x2, y2;
    point2 p2;

    /*centers()*/
    for( i=0; i<numSlices; i++) {
        projection(&slice_center[i], &p2, i);
        x1 = -2 + (int) p2.x;
        x2 = x1 + 4;
        y1 = y2 = (int) p2.y;
        XDrawLine(ncdisplay,ncwindow,ncgc, x1, y1, x2, y2);
        x1 += 2; x2 -= 2;
        y1 -= 2; y2 += 2;
        XDrawLine(ncdisplay,ncwindow,ncgc, x1, y1, x2, y2);
    }

    dispRegion(regno)
    int regno;
    {
        int i;

        if( regno== -1 ) return;

        /*
        switch( regno ) {
            case R_ARM_L:
                region_arm(L);
                break;
            case R_ARM_R:
                region_arm(R);
                break;
            case R_LEG_L:
                region_leg(L);
                break;
            case R_LEG_R:
                region_leg(R);
                break;
            case R_TORSO:
                region_torso();
                break;
            case R_U_TORSO:
                region_up_torso();
                break;
        }
        */
    }
}

break;
case -1:
    return;
default:
    printf("Not a valid choice.\n");
}

for( i=0; i<region_des.nslices; i++ ) highlight(cregion_des.slices[i]);

}

/*
*****
* High light a slice
*****
highlight(sliceNo)
int sliceNo;
{
    int slice_num_points;
    int slice_st,slice_end;

    slice_st = allSlices[sliceNo];
    slice_end = allSlices[sliceNo+1] - 1;
    slice_num_points = slice_end - slice_st + 1;

    XDrawLines(ncdisplay,ncwindow,ncgc,&scr_coors[slice_st],slice_num_points,CoordWo
deOrigin);
    XDrawLine(ncdisplay,ncwindow,ncgc,scr_coors[slice_end].x,scr_coors[slice_end].y
,
    scr_coors[slice_st].x,scr_coors[slice_st].y);
}

highlight_point(pointNo)
int pointNo;
{
    XDrawLine(ncdisplay,ncwindow,ncgc,scr_coors[pointNo].x-2, scr_coors[pointNo].y,
scr_coors[pointNo].x+2, scr_coors[pointNo].y);
    XDrawLine(ncdisplay,ncwindow,ncgc,scr_coors[pointNo].x, scr_coors[pointNo].y-2,
scr_coors[pointNo].x, scr_coors[pointNo].y+2);
    XFlush( ncdisplay );
}

highlight_line(p1,p2)
int p1,p2;
{
    if( p1<0 || p2<0 ) return(0);
    XDrawLine(ncdisplay,ncwindow,ncgc,scr_coors[p1].x, scr_coors[p1].y, scr_coors[p
2].x, scr_coors[p2].y);
    XFlush( ncdisplay );
}

highlight_curve(curve)
curve_dis *curve;
{
    int i;
    int p1, p2;

    for( i=0; i<curve->length; i++ ){
        p1 = curve->path[i];
        p2 = curve->path[i+1];
        XDrawLine(ncdisplay,ncwindow,ncgc,scr_coors[p1].x, scr_coors[p1].y, scr
_coors[p2].x, scr_coors[p2].y);
    }
}

```

```

    }
    XFlush( ncdisplay );

    highlight_vertical_slice(slice)
    disp_pts *slice;
    {
        int i;
        if (is_vert_slice)
        {
            /*
            for (i=slice->num-1;i>0;i--)
                XDrawLine(ncdisplay,ncwindow,ncgc,
                    scr_coords[slice->points[i]].x, scr_coords[slice->points[i]].y,
                    scr_coords[slice->points[i-2]].x, scr_coords[slice->points[i-2]].y);

            XDrawLine(ncdisplay,ncwindow,ncgc,
                scr_coords[slice->points[0]].x, scr_coords[slice->points[0]].y,
                scr_coords[slice->points[1]].x, scr_coords[slice->points[1]].y);

            for (i=1;i<slice->num-2;i++)
                XDrawLine(ncdisplay,ncwindow,ncgc,
                    scr_coords[slice->points[i]].x, scr_coords[slice->points[i]].y,
                    scr_coords[slice->points[i+2]].x, scr_coords[slice->points[i+2]].y);

            */
            for (i=0;i<slice->num;i++)
                highlight_point(slice->points[i]);
        }

        show_vert_slice(vert_slice,axis)
        disp_pts *vert_slice;
        axis;
        {
            int i;
            point3 tmp_pt;
            point2 apt2;
            double angle1,angle2;

            highlight_vertical_slice(vert_slice);

            XClearWindow( ncdisplay, view_win );
            angle1 = 75.0*PI/180.0;
            angle2 = -25.0*PI/180.0;

            for (i=0;i<vert_slice->num;i++) {
                tmp_pt.x = showPoints[vert_slice->points[i]].x;
                tmp_pt.y = showPoints[vert_slice->points[i]].y;
                tmp_pt.z = showPoints[vert_slice->points[i]].z;
                if ( axis == Y_AXIS)
                    rotate( Y_AXIS, angle1, &tmp_pt, &rotOrigin );
                if ( axis == Z_AXIS)
                    rotate( Y_AXIS, angle2, &tmp_pt, &rotOrigin );
                projection( &tmp_pt, &apt2, 0);
                /*
                fprintf(stderr,"x = %d, y = %d\n", (int)apt2.x, (int)apt2.y );
            */
            show_point( (int)((apt2.x/1.65-45.0)), (int)((apt2.y/1.65+40.0))
        }
    }

    show_point( (int)((apt2.x/2.0)), (int)((apt2.y/2.0+30.0)) );
}

```

```

/*****
 * Routines for 3D manipulations.
 * These routines are application independent.
 *****/

#include "transform.h"

double rotateUnit;
int transUnit;
double scaleUnit;

/*****
Project a World Coordinate point to Screen
Coordinate point using Cabinet projection.
*****/
projection(pnt3, pnt2, i)
point3 *pnt3;
point2 *pnt2;
int i;
{
    double length, ratio;

    length = pnt3->z / 2.0;
    ratio = spSize / STD_SP_SIZE; /* Ratio between WCS to SCS */
    pnt2->x = (pnt3->x - length*cos30) * ratio + cop.x;
    pnt2->y = (length*0.5 - pnt3->y) * ratio + cop.y;
}

/*****
Scaling.
*****/
scale(scaleUnit)
double scaleUnit;
{
    spSize += scaleUnit;
}

/*****
Translation around a certain axis.
*****/
translate(anAxis)
short anAxis;
{
    switch (anAxis) {
        case X_AXIS :
            cop.x += transUnit;
            break;
        case Y_AXIS :
            cop.y -= transUnit;
            break;
        case Z_AXIS :
            cop.x -= transUnit*cos30;
            cop.y += transUnit*0.5;
            break;
    }
}

/*****
Rotate a 3D point around a certain axis of the origin.
*****/
rotate(anAxis, angle, point, origin)
short anAxis;
double angle;
{
    point3 *point;
    point3 *origin;
    {
        point2 pnt2;

        switch (anAxis) {
            case X_AXIS :
                pnt2.x = point->y - origin->y;
                pnt2.y = point->z - origin->z;
                rotate2D(&pnt2, angle);
                point->y = pnt2.x + origin->y;
                point->z = pnt2.y + origin->z;
                break;
            case Y_AXIS :
                pnt2.x = point->z - origin->z;
                pnt2.y = point->x - origin->x;
                rotate2D(&pnt2, angle);
                point->z = pnt2.x + origin->z;
                point->x = pnt2.y + origin->x;
                break;
            case Z_AXIS :
                pnt2.x = point->x - origin->x;
                pnt2.y = point->y - origin->y;
                rotate2D(&pnt2, angle);
                point->x = pnt2.x + origin->x;
                point->y = pnt2.y + origin->y;
                break;
        }
    }
}

/*****
This performs a 2D rotation around the 2D origin (0,0)
at a specified degree.
*****/
rotate2D(point, degree)
point2 *point;
double degree;
{
    double length;
    double newAngle;
    double incAngle;

    if( (point->x == 0.0) && (point->y == 0.0) ) return(0);

    length = sqrt( point->x*point->x + point->y*point->y);
    if( incFlag == INCREASE ) incAngle = degree;
    else incAngle = (-1.0)*degree;

    if( point->y > 0.0 ) newAngle = acos(point->x / length);
    else newAngle = pi*2.0 - acos(point->x / length);
    newAngle += incAngle;
    point->x = length*cos(newAngle);
    point->y = length*sin(newAngle);
}

```

Appendix E

Student thesis, Knight

THE MARKET FEASIBILITY OF BODY SCANNING AND
SIZE PREDICTION TECHNOLOGIES AT RETAIL

by

Audra Lynn Knight

A Thesis Submitted to
the Faculty of the Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro 1994

KNIGHT, AUDRA LYNN, M.S. The Market Feasibility of Body Scanning and Size Prediction Technologies at Retail. (1994) Directed by Dr. Nancy L. Cassill. 103pp.

The purpose of this study was to determine the feasibility of a body scanning system for consumer use at retail. Specifically, the appeal and use of three body scan applications were examined: made-to-measure, data card, and computer imaging. The sample chosen for this study consisted of 200 employed female consumers at the University of North Carolina at Greensboro. The survey research method provided a descriptive and an analytical study. A five-page questionnaire was developed by the researcher. This questionnaire was mailed to the sample, along with a cover letter in February 1994.

Descriptive statistics and inferential statistics were used in this study. Means and frequencies were calculated on all items. Analysis of variance (ANOVA) was used to test Hypothesis 1-4. A chi-square test was used to test Hypothesis 5.

Results indicated that the majority of the sample was willing to have a body scan. Neither age nor size had an effect on the appeal or use of the three body scan applications. However, fit problems did have an effect on the appeal and potential usage; those with fit problems found body scanning very appealing. The most appealing and usable application was the data card, followed by the computer

imaging. The made-to-measure application was the least appealing and usable application. Consumers would use the three body scan applications when purchasing bottoms, swimwear, and tops. Consumers were also willing to pay both \$5 and \$10 for an initial as well as an update body scan.

The results of this study have implications for educators, retailers and manufacturers. For educators, this study tested a portion of the Engel, Blackwell, and Miniard decision process model, that of need recognition, and contributed to the void in the literature pertaining to the market feasibility of body scanning and size prediction technologies at retail. Results from this study will provide retailers and manufacturers with a basis to consider body scanning technologies, especially the data card and computer imaging, and adopt body scanning for bottoms, swimwear and tops.

THE MARKET FEASIBILITY OF BODY SCANNING AND
SIZE PREDICTION TECHNOLOGIES AT RETAIL

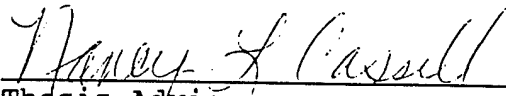
by

Audra Lynn Knight

A Thesis Submitted to
the Faculty of the Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro 1994

Approved by


Thesis Advisor

APPROVAL PAGE

This thesis has been approved by the following committee
of the Faculty of the Graduate School at the University of
North Carolina at Greensboro.

Thesis Advisor Harriet L. Caspell
Committee Members Walter J. Jeffers
David H. Jones

April 18, 1994
Date of Acceptance by Committee

April 18, 1994
Date of Final Oral Examination

ACKNOWLEDGEMENTS

Sincere appreciation and thanks to Dr. Nancy Cassill, for her infinite patience and guidance throughout this study. Greatest appreciation is also expressed to Dr. David Herr and Dr. Maureen Grasso for their interest and valuable assistance as members of my committee.

TABLE OF CONTENTS

	Page
APPROVAL PAGE.	ii
ACKNOWLEDGEMENTS.....	iii
LIST OF TABLES.....	vi
LIST OF FIGURES.....	vii
CHAPTER	
I. INTRODUCTION.....	1
Background of the Problem.....	1
Statement of the Problem.....	3
Significance of the Study.....	3
Limitations of the Study.....	4
Assumptions.....	4
Definition of Terms.....	4
II. REVIEW OF LITERATURE.....	7
Conceptual Model.....	7
Sizing.....	9
Body Scanning.....	12
Female Consumers'/Marketing Issues.....	14
Technology at Retail.....	17
Retailers and Manufacturers.....	19
III. METHODOLOGY.....	21
Research Design.....	21
Hypotheses.....	21
Instrument.....	23
Sample Selection.....	25
Field Test.....	26
Data Collection.....	27
Data Analysis.....	27
Operational Definitions.....	29
IV. RESULTS.....	30
Sample Characteristics.....	30
Research Objective 1.....	31
Research Objective 2.....	37

Testing of Hypotheses.....	38
V. SUMMARY, CONCLUSIONS, & RECOMMENDATIONS.....	46
Summary.....	46
Testing of Hypotheses.....	49
Conclusions.....	50
Implications.....	52
Recommendations.....	52
BIBLIOGRAPHY.....	54
APPENDICES.....	58
A COVER LETTER, QUESTIONNAIRE,.....	58
& HUMAN SUBJECTS APPROVAL FORM	
B SAMPLE CHARACTERISTICS.....	66
C CHI SQUARE ANALYSIS.....	83
D ANALYSIS OF VARIANCE AND MEANS.....	94

LIST OF TABLES

Table		Page
1	Appeal and Usage of Body Scanning.....	33
2	Appeal of Body Scanning.....	34
3	Use of Body Scanning.....	34
Appendix		
B-1	Description of Respondents.....	67
B-2	Description of Respondents (Means).....	78
B-3	Vector Analysis (Appeal).....	79
B-4	Vector Analysis (Use).....	81
C-1	Chi-Square Test-Appeal of (a) by Use of (a)....	84
C-2	Chi-Square Test-Appeal of (b) by Use of (b)....	85
C-3	Chi-Square Test-Appeal of (c) by Use of (c)....	86
C-4	Chi-Square Test-Pay for Initial Scan by Pay.... for Update	87
C-5	Chi-Square Test-Fit Problems by Pay for..... Initial Scan	88
C-6	Chi-Square Test-Fit Problems by Pay for..... Update Scan	89
C-7	Chi-Square Test-Age by Most Appealing.....	90
C-8	Chi-Square Test-Age by Most Usable.....	91
C-9	Chi-Square Test-Body Size by Most Appealing....	92
C-10	Chi-Square Test-Body Size by Most Usable.....	93
C-11	Chi-Square Test-Fit Problems by Amount Spent... on Wardrobe	94

LIST OF FIGURES

Figure		Page
1	EBM Decision Process Model.....	8

CHAPTER I

INTRODUCTION

Background of the Problem

Retailers in the 1990's are challenged to provide products and services to meet consumers' apparel needs. One consumer need is the fit of apparel products which involves "fitting" an apparel product on the three-dimensional (3-D) human body, consisting of the dimensions of depth, width and height. Historically, ready-to-wear apparel has been created by designers preparing two dimensional "design concepts" which treat the human body as if it were flat and divided into separate unconnected segments (Martell, 1990). The resulting products are designed for the general mass market, not individuals. Proper fit remains a critical issue in customer satisfaction of apparel products.

The apparel industry provides garments in many sizes which must fit the 3-D human body. Sizing standards have assigned fixed dimensions to each size based on the assumption that the majority of people within each size would have the same standard measurements or size specifications. However, the standards are based on data collected decades ago and are no longer applicable to current needs (Tamburrino, 1992). The current sizing standards are not suitable for all body types and have been shown to cause lower body cathexis in consumers

who do not fit the "ideal" body type. The latest generation of electronic machines makes it possible to create garment shapes that are 3-D and incorporate precise human measurements (Gray, 1994).

Sizing methods for men are different than those for women. Current sizing for women is not reliable for the industry nor the consumer. It is estimated that 70-80% of the garments in stores may not fit the size stated. According to DeLong, Ashdown, Butterfield & Turnbladh (1993), the development of a system that would accommodate individual body measurements, configurations, alignments, and proportions at a reasonable cost would assist consumers in fitting products. At this time, sizing is a major problem for all consumers.

To address this problem, sizing studies are being conducted to aid manufacturers in creating better fitting garments. At this point, if a consumer desires custom fit apparel they have no choice but to turn to made-to-measure apparel. However, new technology is being developed to provide body scanning and size prediction services to the average consumer. Body scanning is a computerized body measurement system designed to take accurate measurements of the human body ("Textile/Clothing", 1993). Size prediction is the computer software to convert 3-D "frame drawings" into size information for specific brands (Martell, 1990). The body scanning and size prediction technology is in the final

stages of development and is expected to appear in the marketplace by the end of 1994. Three outputs of the body scanning technology are (a) made-to-measure, (b) data card, and (c) computer imaging.

To date there has been no consumer research on the market feasibility of body scanning. In the consumer driven marketplace, it is important to understand the consumer's current problems with apparel products and their resulting attitudes and beliefs in order to design and implement a system to meet consumer needs.

Statement of the Problem

The purpose of this study is to determine the feasibility of a body scanning system for consumer use at retail. The four specific research objectives are:

- 1) To determine consumers' interest in body scanning. Specifically, determine consumers' acceptability of three body scan outputs: made-to-measure, data card, and computer imaging.
- 2) To assess current sizing problems with the fit of apparel.
- 3) To determine if acceptability differs by demographics.
- 4) To determine if fit problems differ by demographics.

Significance of the Study

The results of this study will benefit consumers, retailers, and manufacturers. Consumers will be exposed to

the body scanning process and will have the opportunity to provide input regarding usage of the technology, as well as convey consumer fit problems to the manufacturer. For retailers, results will help determine if there is an interest in body scanning in specific markets and the feasibility of purchasing a body scan system. Manufacturers will also benefit from this information by becoming more knowledgeable about the fit problems of consumers and the feasibility of such systems. The results of this study will be used to help design the second research phase, which will involve consumer's use of body scanning technology and the third phase, which will determine retail feasibility of body scanning and size prediction technologies.

Limitations of the Study

- 1) The random sample is limited to female State Personnel Act (SPA) employees at the University of North Carolina at Greensboro.

Assumptions

- 1) Apparel purchasers have problems with the fit and sizing of garments.
- 2) Body scanning technology is virtually unknown to most consumers.

Definition of Terms

Acceptability

Consists of two components: appeal and usage.

Apparel Product	General term that includes men's, women's, and children's clothing (Jarnow & Guerreiro, 1991).
Body Cathexis	The evaluative dimension of body image, defined as positive and negative feelings toward one's body (LaBat & DeLong, 1990).
Body Scanning	A computerized body measurement system designed to take accurate measurements of the human body with p h o t o l i g h t c e l l s ("Textile/Clothing", 1993).
Body scan outputs	Consists of three applications (a) made-to-measure, (b) data card, and (c) computer imaging.
Custom Fit	Apparel made to the order of individual customers; cut and fitted to individual measurements (Jarnow & Guerreiro, 1991).
Made-to-Measure	Clothing manufactured specifically for an individual to one's measurements (Oliver, Bickle, & Shim, 1993).
Ready-to-Wear	Apparel that is mass produced in standard sizes (Jarnow & Guerreiro, 1991).
Scanning	Facilitates data entry and capture through optical reading of these into the unique item numbers and other information ("Quick Response", 1991).
Size Prediction	Computer software designed to convert 3-D "wire frame drawings" into size information for specific brands (Martell, 1990).
Size Specifications	The body dimensions suitable for the labeled size of a garment (Brown, 1992).

Three-Dimensional (3-D) Having, or seeming to have the dimensions of depth as well as width and height (Woolf, 1976).

CHAPTER II

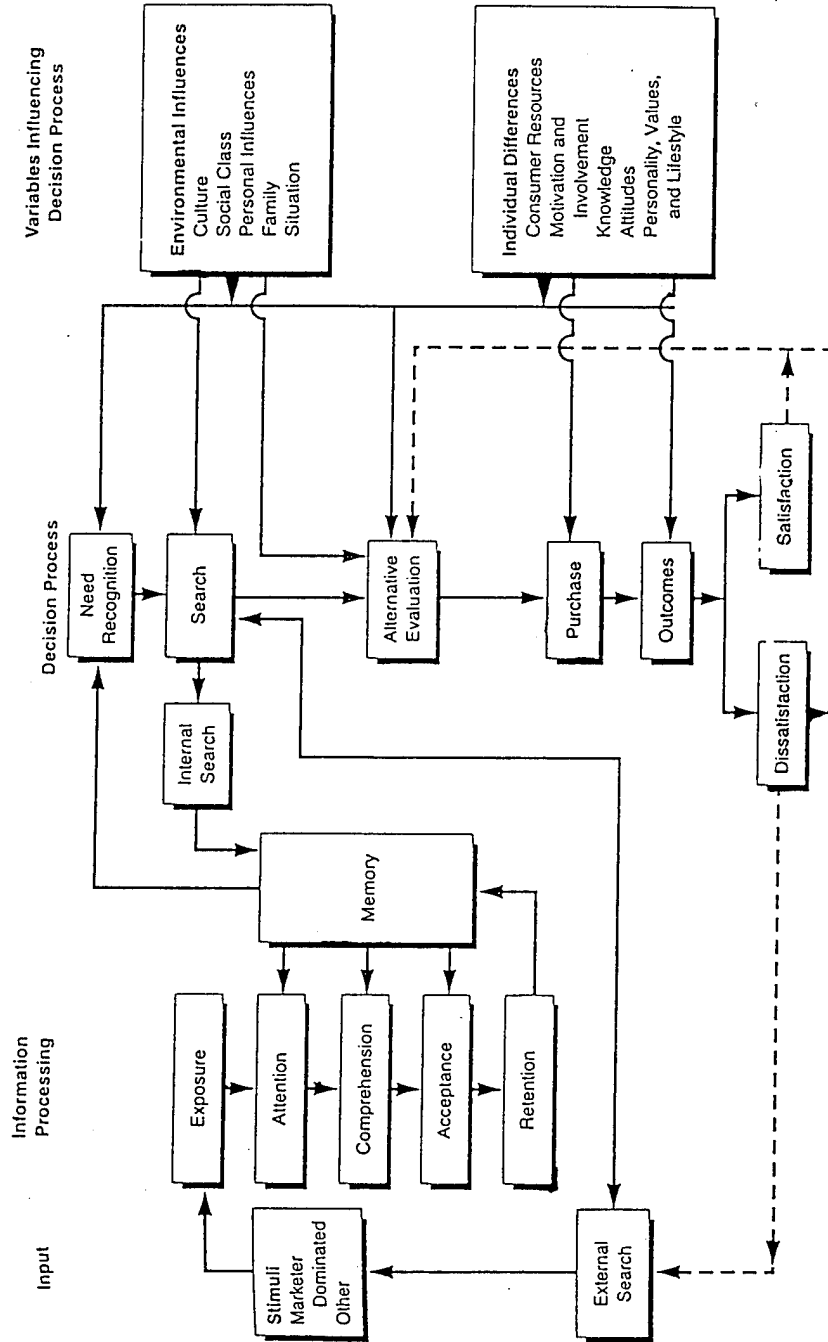
REVIEW OF THE LITERATURE

The review of the literature is separated into five sections: 1) the conceptual model, 2) sizing issues, 3) body scanning, 4) female consumers marketing issues, and 5) retailing and manufacturing.

Conceptual Model

The Engel, Blackwell, and Miniard (EBM) Decision Process Model (Engel, Blackwell & Miniard, 1993) is the conceptual model used in this study (See Figure 1). This model identifies the various components and thought processes that are used by consumers in the decision making process. The EBM model illustrates that environmental and individual influences as well as market-oriented influences impact the decision making process.

The decision process contains five stages: need recognition, search, alternative evaluation, purchase, and outcomes. The model suggests that consumers follow this five-stage process when they are selecting a product. The arrows which connect the boxes (--->) indicate how the various stages influence the purchase decision. The focus in this study is on the first stage: need recognition.



A Complete Model of Consumer Behavior Showing Purchase and Its Outcomes

Reference: Engel, J.F., Blackwell, R.D., & Miniard, P.W. (1993).
Consumer behavior (7th ed.). Chicago, IL: The
 Dryden Press, p. 53.

Need Recognition

Need recognition is the first stage in the decision process. Need recognition essentially depends on how much discrepancy exists between the actual state (i.e., the consumer's current situation) and the desired state (i.e., the desired situation). When this discrepancy exceeds a certain level or threshold, a need is recognized (Engel, Blackwell, & Miniard, 1993).

Marketers are unable to create needs for consumers but can influence them by activating needs that already exist within consumers. Product innovations are a source of need recognition that can be used by marketers (Engel, Blackwell, & Miniard, 1993). An approach to uncovering need recognition is to measure the attitude/behavior relationship. The innovative design of body scanning could make the traditional sizing standards used by apparel manufacturers seem obsolete to consumers by providing an entirely different concept of what is now possible. Need recognition leads to the second stage of the decision process, information search.

Sizing

Presently, ready-to-wear apparel consumers discover their size by trying on garments. This process can be very time consuming considering sizes vary between styles, designers and manufacturers. Consumers whose body types differ from the prescribed standards must choose the most satisfactory size

from those available. It is apparent that the sizing system based on ideal proportion is too limited and there is no knowledge to say this differs by age. A departure from stereotypical definitions of female body types could result in new sizing systems that would fit more consumers (LaBat & DeLong, 1990). At this time, those consumers who cannot find styling or sizing in the ready-to-wear market have to look to custom fit apparel (DeLong, Ashdown, Butterfield & Turnbladh, 1993).

The sizing systems used by the apparel industry are based on the ideal. According to LaBat et al. (1990), the current systems provide a symbol of expectation for women. When a woman tries to fit her body with currently available clothes the comparison to the ideal is inevitable. The sized garment that does not fit sends a message to the woman that her body is less than perfect. Women with high body cathexis, positive feelings toward one's body, are not necessarily more satisfied with the physical fit of their ready-to-wear clothing (Shim & Kotsiopulos, 1990). If this is true, it is easy to see why general sizing and fit problems continue to be a source of frustration for those consumers who are not considered "average."

When consumers are considering apparel for purchase, fit must be minimally satisfactory. When clothing does not fit, the consumer may perceive the cause as related to the body and

not the clothing. This results in negative feelings toward the body. Body cathexis is the evaluative dimension of body image and is defined as positive and negative feelings toward one's body (LaBat & DeLong, 1990).

LaBat (1988) reported that the higher the body cathexis, the more satisfied female consumers were with the physical fit of clothing. In a study correlating body cathexis and satisfaction with the fit of apparel, petite women were least satisfied with their bodies and height and were most dissatisfied with sizing and fit of apparel compared to average women (Shim & Kotsiopoulos, 1990).

Petite women have indicated several fit problems in purchasing apparel products (Wallach, 1986). Apparel manufacturers have not recognized the actual problems and needs of these consumers. Shim and Kotsiopoulos (1990) found that average-sized women were most satisfied with the number of stores that carry their size, the size range available, and the general fit of clothing, but that petite women are most dissatisfied with these three items.

Large-sized women also have problems with the fit of apparel. Chowdary and Beale (1988) reported that large-sized women were dissatisfied with apparel sizing and fit. DuCoffe and Cohen (1980) identified buying clothes as the worst problem of large-sized women. Deonier, DeLong, and Martin (1979) emphasized reevaluation of sizing standards and procedures to improve fit for large-sized individuals.

In investigating men's body size and clothing, Shim, Kotsiopulos, and Knoll (1990) reported that short men showed lower body cathexis than big, tall and average-height men. Short and big men were most dissatisfied with size ranges available in ready-to wear. Size and fit have been found to be the most common problems with suits for big and tall men (Shim & Kotsiopulos, 1991). As a result, many men have turned to made-to measure clothing which may offer improved fit over ready-to-wear for persons who are short, tall, and big because it takes into consideration the individual needs of each consumer (Oliver, Bickle & Shim, 1993).

The made-to-measure clothing business, defined by the apparel industry as clothing specifically designed for an individual to one's own measurements, is flourishing in the U.S. (Oliver et al., 1993). This custom made clothing offers improved fit because individual needs are considered. Made-to-measure is characterized by convenient service and selection. Oliver et al. (1993) reported that the price of menswear made-to-measure garments are comparable to ready-to-wear tailored garments at specialty stores.

Body Scanning

Body scanning is a computerized body measurement system designed to take accurate measurements of the human body with photo light cells ("Textile/Clothing", 1993). Recent advances in technology have led to the development of a new non-contact

body measurement system that can measure the 3-D body both quickly and accurately. The 3-D data can be used to create a 3-D model of the body. Special image rendering software has also been created to produce realistic images of what the 3-D model would look like under varying lighting conditions (Martell, 1990).

Other industries have developed similar 3-D scanning systems to insure proper sizing and fit. The footwear industry has employed the use of 3-D computer aided design (CAD) systems for years. Light-sensitive photo cells are being used to scan customers feet from the side and bottom to acquire the information needed to make a custom-fitted shoe. The process is simple; the customer steps into a scanner, pushes a button and measurement of the right and left foot are sent directly to a CAD system where a laser cuts the leather. The shoes are made and mailed within five days. A second application of the 3-D scanning system is being used to fit the human body to a bicycle frame. Custom measurements are used as a "fit kit" in the design (Harris, Mehrman & Dougherty, 1992). In addition to being used in the shoe and bicycle industries, body scanning will be used in the apparel industry to produce better fitting garments.

Technology, one potential solution for sizing challenges, is being developed by (TC)² and the Clemson Apparel Research Center. This technology, known as the body scanner, can

accurately scan the human body with a new non-contact body measurement system and is expected to be available at the end of 1994. Software that accompanies this technology is used to read 1) body measurements from the scan for made-to-measure apparel and 2) predict sizes for ready-to-wear apparel. The system is specially designed to take accurate measurements of the body in a computerized booth in six seconds or less. The output of the body scan can then be used to: 1) manufacture made-to-measure clothing, 2) generate a "data card" with all the consumers body measurements, and/or 3) superimpose garments on the computer image of the consumer's body.

The body scanning technology will be available to consumers in the retail setting. To date no consumer research exists to test the acceptability of such a system for consumers in the retail setting.

Female Consumers'/Marketing Issues

Female consumers are recognized by marketers as primary purchasing agents of consumer products such as apparel (Cassill & Drake, 1987). In 1990, the average household's annual expenditure for apparel was \$585 (Waldrop & Mogelonsky, 1992). The International Labor Office (1992) has shown that fifty-five percent of all women are in the labor force in the United States; 65% of younger women, ages eighteen to thirty-four are employed, compared with 70% of middle-aged women (thirty-five to fifty-four) and 20% of older

women (fifty-five and over) (Waldrop & Mogelonsky, 1992). The breakdown of the female labor force participation rate by ethnic background (Person, 1993) is as follows: 56% white, 58% black, 53% Hispanic and 56% of Asian women.

Women dominate the service-producing industries supplying 80.5% of the labor force in these segments. The top four employment sectors for women are: 1) professional and related services, 2) retail trade, 3) manufacturing, and 4) finance, insurance and real estate (Shortridge, 1987).

Middle-aged women are 20% more likely than other women to purchase a suit each year, and they are 40% more likely to buy a blazer. Sixty percent of middle-aged women buy four or more dresses a year, compared with 40% of younger women and fewer than 30% of older women. Women aged 35 to 54 also pay more for most of the items they buy (Waldrop & Mogelonsky, 1992).

Employed women's apparel consumption patterns differ from nonemployed women. Employed women place greater value on time-saving, convenience-shopping outlets, place greater accent on fashion, take considerable interest in clothing's flattering qualities and its suitability for work, and spend more on clothing (Shim & Drake, 1988). Working women are more than twice as likely to spend \$500 or more on apparel per year than non-employed women (O'Hare, 1993).

Many female consumers seek information about apparel products from the retail store. A study by Kerin, Jain and

Howard (1992), found that the consumer's information search process is impacted by the interaction with a store's physical surrounding. The pleasantness or unpleasantness of a store can influence the patronage decisions and purchase intentions.

Many employed female consumers are influenced by personal sources when searching for apparel. Solomon (1985) reported that career women obtain most of their information on career dress from women friends and coworkers. A study by Cassill and Drake (1987) found that working women who were classified as having "just a job" sought information about apparel products from friends. Shim and Drake (1989) identified the "pal advice searcher", the segment of the population as a woman most likely to talk to friends, colleagues and family about new clothes during the information search process.

Female consumers also use non-personal sources when searching for apparel. Research by Shim and Drake (1989), identified female consumers who gained information from various print sources including business magazines, advertisements and newspaper articles with information on employment clothing.

Research by Shim and Kotsiopoulos classified female apparel shoppers into three categories: "highly involved", "convenience oriented catalog" and "apathetics". They found that the "highly involved" shoppers used fashion publications and mass media moderately during search, but that convenience-oriented catalog shoppers most frequently used fashion

publications. Body scanning is a non-personal source that can aid female consumers in the search process for apparel products.

Technology at Retail

Consumers are becoming increasingly familiar with technology in the retail setting. Some of these technological advances are experienced directly by the consumer, such as the kiosk. Other advances are used for efficiency between the retailer and manufacturer, such as barcoding and scanning, and effect the consumer indirectly.

Innovative retailers are already providing consumers with new and exciting ways to shop. One of the fastest growing trends in the industry is the kiosk. A kiosk is a "cabinet enclosed" interactive computer that uses touch, video and sound to supply information and sell products (Chandler, 1992). Kiosks are already found at such locations as stores, malls, college campuses, offices and airports.

A kiosk is used to make buying decisions more quickly and conveniently. A number of leading retailers are using electronic kiosks to offer customers expanded merchandise assortments not normally stocked in their stores. Consumers can view and compare products, obtain product information, determine whether the product is in stock, and even order and pay for the product at the kiosk.

Retail stores with kiosks in place attribute the acceptance to consumers increased comfort with computers. One ATM-like kiosk has many capabilities. Consumers are using kiosks to obtain coupons with their personal "frequent shopper" cards (Rowland, 1990). Dayton Hudson Corporation has a kiosk program that offers complete bill payment services. Other kiosk programs offer personalized gift certificates and bridal registries that allow the bride and groom to walk around the store with a scanner and record their wish list into the system ("Touch Screen", 1994).

Consumers are using other forms of technology at retail to make their purchase process easier. Target Greatland has installed scanners and telephones throughout its stores. The scanners allow shoppers to perform their own price checks, while the telephones connect shoppers to the service desk to ask questions or request personal service (Jacobs, 1994).

Technology is being utilized to improve the relationships and procedures that speed the flow of information and merchandise between retailers and manufacturers. Barcoding and point-of-sale (POS) scanning are two such technologies that allow retailers to 1) increase customer checkout productivity by eliminating manual entry of information, 2) track merchandise at the item level throughout the pipeline, 3) eliminate the need to re-mark merchandise for promotions, 4) increase distribution center productivity and speed by

reducing or eliminating manual receiving and checking procedures ("Quick Response", 1991). The industry-wide use of scanning will continue to grow because consumers are accustomed to seeing POS systems when they shop ("C-Stores", 1994). These technologies have proven to be good investments in the retail community for Macy's Northeast and Wal-Mart (Robins, 1992).

Retailers and Manufacturers

The retail industry is responsible for the selling of goods and services to the ultimate consumer (Morganstein & Strongin, 1987). The industry consists of 135,000 retailers specializing in fashion apparel and accessories in their merchandise assortment (Jarnow & Guerreiro, 1991). In 1993, it was estimated that 19,346,000 people were employed in the retail industry (U.S. Department of Commerce, 1993).

The domestic apparel manufacturing industry is composed of firms that are vertically integrated to varying degrees. Some companies perform all activities from design to production to distribution of branded apparel. Other companies focus on cutting the piece goods and sewing the garments, but do not design apparel, purchase the raw materials, or distribute the finished goods. Still others license designs from independent designers then manufacture and distribute the garments (Troxell, 1976).

More than one million people are employed in the U.S. apparel manufacturing industries; 842,000 are production workers. There are 21,301 apparel or other textile product companies with 23,168 plants in the U.S. (Grill & Sharkley, 1991). Retailers and manufacturers have realized that developing strong, mutually interdependent relationships is critical in achieving consumer satisfaction. The heart of retailer-manufacturer relationships is the sharing of information through the use of technology. Retailers are sharing point-of-sale (POS) scanner data with suppliers and receive automatic replenishment from suppliers thereby eliminating the need for large inventories.

CHAPTER III

METHODOLOGY

The methodology chapter is divided into eight sections: 1) research design, 2) hypotheses, 3) instrument, 4) sample selection, 5) field test, 6) data collection, 7) data analysis, and 8) operational definitions.

Research Design

The survey research method provides a descriptive and an analytical study. The descriptive study determines consumers' acceptability of body scanning, specifically the three body scanning outputs: made-to-measure (application a), data card (application b), and computer imaging (application c) and gives a current understanding of sizing problems with the fit of apparel for female consumers (research objectives 1 & 2). The analytical study examines differences in acceptability by demographic variables (research objective 3, hypotheses 1-4). The analytical study also examines differences in fit problems by a demographic variable (research objective 4, hypothesis 5).

Hypotheses

The statistical and alternative hypotheses tested in this study are given below. The research hypotheses are the same as the statistical alternative hypotheses in each case.

H1A: Age has no effect on the appeal of application (a).

H1B: Age has no effect on the appeal of application (b).

H1C: Age has no effect on the appeal of application (c).

A1A: Age has an effect on the appeal of application
(a).

A1B: Age has an effect on the appeal of application
(b).

A1C: Age has an effect on the appeal of application
(c).

H2A: Age has no an effect on the use of application (a).

H2B: Age has no an effect on the use of application (b).

H2C: Age has no an effect on the use of application (c).

A2A: Age has an effect on the use of application
(a).

A2B: Age has an effect on the use of application
(b).

A2C: Age has an effect on the use of application
(c).

H3A: Body size has no effect on the appeal of application (a).

H3B: Body size has no effect on the appeal of application (b).

H3C: Body size has no effect on the appeal of application (c).

A3A: Body size has an effect on the appeal of
application (a).

A3B: Body size has an effect on the appeal of
application (b).

A3C: Body size has an effect on the appeal of
application (c).

H4A: Body size has no effect on the use of application (a).

H4B: Body size has no effect on the use of application (b).

H4C: Body size has no effect on the use of application (c).

A4A: Body size has an effect on the use of application (a).

A4B: Body size has an effect on the use of application (b).

A4C: Body size has an effect on the use of application (c).

H5: There is no association between amount spent on wardrobe and fit problems.

A5: There is an association between amount spent on wardrobe and fit problems.

Instrument

A five-page questionnaire (See Appendix A) was developed to collect data with the assistance of Dr. Nancy Staples, Clemson Apparel Research Center. Industry interviews were conducted with (TC)² and Wrangler for further input in the questionnaire. This instrument was refined with the help of Dr. David Herr, a UNCG Statistical Consulting Center statistician.

The instrument consisted of four sections of forced-choice type questions to address the research objectives and the hypotheses. Section I included a two paragraph description of a body scan experience followed by five

questions related to this experience. One question (Q#1) assessed a consumers' willingness to have a body scan. Question #2 indicated which price the consumer was willing to pay for a body scan initially and for an update. How accessible a body scanner has to be in order for the consumer to use it was assessed by Question #3. Question #4 determined whether the consumer would be more willing to have a body scan if a body suit was worn. Question #5 asked if the consumer thought having a body scan was worth their time in order to have better fitting apparel.

Section II included the description of three different applications of the body scan output and asked questions about the appeal and usage of body scanning. The appeal of the three applications (Q#6,7,8) consisted of a Likert scale (1=Not at all, 5=Very) with Question #9 asking which application was most appealing. Questions #10, #11, and #12 asked the consumer how often they would use the three applications (Likert scale, 1=Not at all; 5=Very). Question #13 determined which application would be used most frequently. Question #14 asked for which of eight apparel products would the consumer most likely use body scanning. Appeal and usage will be used to measure acceptability.

Section III consisted of questions relating to the fit of everyday apparel and jeans including the fit of tops and bottoms (Q#15). Question #16 asked whether the consumer tries

on several size garments before one is found that fits. Two questions requested the consumer's size when purchasing shirts and blouses (Q#17 Junior, Junior Petite, Missy, Missy Petite, and Womens) and jeans and slacks (Q#18). The following two questions addressed alterations, specifically, frequency of alteration (Q#19), (Likert scale; 1=Never, 3=Always) and consumer's willingness to pay more (Q#20) and dollar amount more (Q#20(1)) for everyday apparel made to their size specifications. Question #21 asked the consumer if they had ever purchased made-to-measure apparel. Questions #22-25 related to consumer fit problems with jeans including length, waist, hips (Q#23); number of garments tried (Q#24), and difficulty in fit (Q#25), (Likert scale; 1=Easy, 5=Difficult). The final question of this section (Q#26) asked questions regarding frequency of alterations (Likert scale; 1=Never, 3=Always).

Finally, Section IV contained demographic questions (Q#27-33) consisting of age, household income level, ethnic background, education level, dollar amount spent on wardrobe last year and computer usage. The last question (Q#33) was an open-ended question that asked if the consumer had anything they would like to tell about the size and/or fit of apparel.

Sample Selection

The population consisted of State Personnel Act (SPA) employees at the University of North Carolina at Greensboro.

The list consisted of 820 people (273 male, 527 female) employed full-time in various positions ranging from research assistants to secretarial to housekeeping. The 273 males were eliminated from the population so that the study would focus exclusively on female consumers, purchasers of the majority of apparel products.

From the remaining 527 females, a sample of 200 was generated through a pseudo-random table of numbers. This was accomplished by numbering the list of 527 females, indicating there were 527 to the computer, drawing 200 numbers from the computer and selecting the corresponding numbers on the list. Permission from the University of North Carolina at Greensboro was obtained to use human subjects in this study (see Appendix A).

Field Test

A field test of the instrument was conducted with a senior-level textile products marketing and a retail buying class at the University of North Carolina at Greensboro. Forty students completed the questionnaire and provided suggestions regarding questionnaire format, ease of understanding, length of time required, and the order of questions. These suggestions were used to refine the questionnaire before it was administered to the sample. Question 2 was changed to distinguish between paying for an initial and updated body scans. After the first field test,

Section I and Section II were reversed because suggestions indicated that a consumer might be more willing to have a body scan after they knew what they could do with the output. The second field test neither supported nor opposed this belief, therefore Section I and Section II were used in the original order.

Data Collection

The survey research method was used to collect data. A cover letter and five-page questionnaire were mailed to the SPA employees via campus mail in February 1994 (Appendix A). The cover letter explained the purpose of the study and asked the respondent to return the completed questionnaire to a campus address by a date which was approximately two weeks after they received the questionnaire. An identification number (1-200) was placed on the back of the fourth page at the bottom right corner for identification purposes only. The purpose of the identification number was clearly explained in the cover letter.

Data Analysis

Means and frequencies were calculated on all items. Descriptive statistics and inferential statistics were used to determine consumers' interest in body scanning and to assess current sizing problems with the fit of apparel (Research Objectives 1 and 2). Univariate procedures were performed to identify differences in appeal and use of applications (a),

(b), and (c). Vector descriptions were used to identify frequent response combinations relating to appeal and use (Q#6-8, Q#10-12; research objective 1). Chi-square tests provided inferential statistics.

To test Hypotheses 1 & 2, the age variable was collapsed into four levels. The first two levels were combined into one group, as were the last two levels, creating four new age groups. Analysis of variance (ANOVA) was used to determine if the four consumer age groups differed on the appeal of body scanning.

To test Hypotheses 3 & 4, a size variable was created (from responses to Q#17 & 18) to identify female consumers who wore small, average, and large sizes. The small group was defined as sizes less than or equal to 7 in Junior, Junior Petite, Missy, and Missy Petite. The average group was defined as sizes greater than or equal to 8 and less than or equal to 12 in Junior, Junior Petite, Missy, and Missy Petite. The large group was defined as 13 or greater in Junior, Junior Petite, Missy, Missy Petite, and Womens sizes. The three size groups were validated by an industry representative. Analysis of variance (ANOVA) was used to determine if the three size groups differed on the appeal of body scanning for each of the three applications.

To test Hypothesis 5, chi-square tests were used to determine if there was an association between amount spent on wardrobe and fit problems (Q#15).

Operational Definitions

Acceptability:	Consists of two components: appeal and use (Continuous, Q#6, Q#7, Q#8 (appeal); Q#10, Q#11, Q#12 (use); Categorical, Q#9, Q#13).
Age:	Respondent's age divided into 4 groups (1) younger than 30, (2) 30-39, (3) 40-49, and (4) 50 and older (Categorical, Q#27).
Amount spent:	The amount spent on wardrobe last year (Categorical, Q#31).
Appeal:	The appeal of the three body scan applications: made-to-measure, data card, and computer imaging (Continuous, Q#6, Q#7, Q#8; Categorical, Q#9).
Body scan application:	Three different outputs of information from a body scan: made-to-measure, data card, and computer imaging (Continuous, Q#6, Q#7, Q#8 (appeal); Q#10, Q#11, Q#12 (use); Categorical, Q#9, Q#13).
Ethnic background:	The race of the consumer, Asian, Black, Hispanic, White (Categorical, Q#29).
Large sizes:	Sizes 13 and greater in junior, junior petite, missy, missy petite, and womens sizes (Categorical, Q#17, Q#18).
Made-to-measure:	Apparel made to one's own size specifications (Categorical, Q#21).
Size:	The name of the category of clothes a female wears to fit her figure type: junior, junior petite, missy, missy petite, and womens sizes (Categorical, Q#17, Q#18).
Use:	The use of the three body scan applications: made-to-measure, data card, and computer imaging (Continuous, Q#10, Q#11, Q#12; Categorical, Q#13).

CHAPTER IV

RESULTS

The purpose of this study was to determine the market feasibility of a body scanning system for consumer use in the retail setting. The four specific research objectives were:

- 1) To determine consumers' interest in body scanning. Specifically, determine consumers' acceptability of three body scan outputs: made-to-measure, data card, and computer imaging.
- 2) To assess current sizing problems with the fit of apparel.
- 3) To determine if acceptability differs by demographics.
- 4) To determine if fit problems differ by demographics.

Sample Characteristics

A total of 97 (out of 200) employed female consumers returned the questionnaire (48.5% response rate). The largest group of respondents (36.1%) were ages 30 to 39. Of the remaining respondents, 16.5% were below age 30, 25.8% were ages 40 to 49 and 19.6% were ages 50 and above, Tables B-1 and B-2.

The household income level of 24.7% of the respondents was below \$30,000 a year. Thirty-one percent of the respondents had an annual household income level between

\$30,000 and \$49,999, followed by 40.2% of the respondents who earned above \$50,000 a year.

The majority of the sample (84.5%) was white, with the remaining sample being black (11.3%) and Asian (1.0%) (no response=3.2%). Approximately half of the respondents (47.4%) had a high school diploma and some college or vocational training, followed by 25.5% of the respondents who held a bachelor's degree. Only 10.3% of the sample had some graduate school, with 14.4% holding graduate degrees (no response=2.4%).

As a whole, 45.4% of the sample spent between \$200 and \$499 on their wardrobe last year and 30% spent between \$500 and \$999. Only 10.3% spent less than \$200; 11.3% spent more than \$1,000 (no response=3.0%). Nearly all of the respondents (97%) used a computer in the work environment.

Research Objective 1

Research objective 1 sought to determine interest in the three body scan outputs: (a) made-to-measure, (b) data card, and (c) computer imaging. Interest was measured with seven components: willingness, appeal, use, relationship of appeal and use, product, willingness to pay, and accessibility.

Willingness

Over three-fourths of the sample (76%) were willing to have a body scan. The majority (67%) of the respondents believed that having a body scan was worth the time.

Appeal

All of the body scan applications were appealing to the respondents (Table 1). Application (b), data card, was most appealing to respondents (33%), followed by application (c) at 32.0% (computer imaging) and application (a) at 21.6% (made-to measure). Vector descriptions (Appendix B, Table B-3) were used to uncover the most common response combinations for appeal. Sixteen consumers indicated that all three applications ((a), (b) and (c)) were very appealing. These consumers can be characterized as having fit problems and having to try on several sizes before finding one that fits. Eleven consumers found all three of the applications not at all appealing, and nine found all of the applications somewhat appealing. The body size of the consumer was not an issue when measuring appeal in this case.

Use

Consumers' response to usage of the body scan applications were not very different from responses to appeal (Table 1). Respondents indicated (37.1%) that they would most use application (b), data card. Application (c), computer imaging, was indicated to be the most used by 32% of respondents. Only 18.6% of respondents said they would most use application (a), made-to-measure (no response=12.3%). Vector descriptions (see Appendix B) were used to uncover the most common response combinations for usage. Ten consumers

indicated that all three applications ((a), (b) and (c)) would not be used. These consumers can be characterized as having very few fit problems with no need to try on several sizes to find the correct fit. The body size of the consumer was not an issue when measuring use in this case.

Table 1
Appeal and Usage of Body Scanning

<u>Application</u>	<u>Means/Std Error</u>	
	<u>Appeal</u>	<u>Usage</u>
(a) Made-to-measure	3.34/1.51	2.75/1.27
(b) Data card	3.59/1.38	3.25/1.35
(c) Computer imaging	3.48/1.47	3.11/1.42

Note: The means value ranged from 1--5 (1=Not at all, 5=Very).

Relationship of appeal and use

The differences in consumers' responses to appeal and usage are presented in Tables 2 and 3. In general, the appeal of the body scanning applications is greater than the usage. To further examine appeal and usage, univariate results indicated the following for each of the three applications: (1) some respondents (36.5-44.8%) found the applications more appealing than usable, (2) almost half (50.0-54.2%) found the applications equally appealing and usable, and (3) a few respondents (5.2-9.3%) found the applications more usable than appealing.

Table 2

Appeal of Body Scanning

Application	Not at all	Somewhat	Very
(a) Made-to-measure	25.8%	27.8%	46.0%
(b) Data card	20.6%	19.6%	58.7%
(c) Computer imaging	26.8%	17.5%	54.6%

Note: Percentages based on sample size of 97.

Table 3

Use of Body Scanning

Application	Not at all	Somewhat	Very
(a) Made-to-measure	42.2%	29.9%	26.8%
(b) Data card	32.0%	17.5%	49.5%
(c) Computer imaging	35.1%	20.6%	43.2%

Note: Percentages based on a sample size of 97.

Chi-square tests were conducted to determine if there was an association between appeal and usage for each of the three applications. In all three cases the chi-square statistic was significant ((a), $P=0.0000$; (b), $P=0.0000$; (c), $P=0.0000$). The chi-square statistic within cells was examined (Appendix C, Table C-1-3). The chi-square statistic for the cell not at all appealing and not often used (1,1) for each of the three applications was greater than expected ((a) 33.607; (b) 59.559; (c) 44.501). The cell very appealing and very often

used (5,5) also had a greater chi-square statistic for each of the three applications ((a) 14.667; (b) 18.002; (c) 11.403). Those respondents that thought the three body scan applications were appealing would use them; those who did not think the three applications were appealing would not use them.

Product

Respondents were asked to choose from a list all products for which they would potentially use body scanning. The results indicated that consumers would use body scanning for jeans and slacks (75.3%), followed by 53.6% for swimwear and 52.6% for blouses and shirts. There was also interest in using body scanning for other apparel products such as skirts, shoes, dresses, suits and coats.

Willingness to pay

When asked how much they would pay for an initial body scan, 36.1% of respondents said \$5 while 20.6% indicated \$10. Respondents (44.3%) were also willing to pay \$5 for an updated body scan. Three percent of respondents would pay \$10 for an updated body scan.

Results from a chi-square test indicated that 38.4% of respondents would not pay for an initial or update body scan (Appendix C, Table C-4). One respondent (1.2%) who was not willing to pay for an initial scan was willing to pay \$5 for an update of the card; no one was willing to pay \$10. The

tests showed that 10.5% of the respondents (n=9) who paid \$5 for an initial scan would not pay for an updated scan; 27.9% would pay another \$5 for an updated scan, but no one was willing to pay \$10 for an updated scan. All those respondents (n=19) who would pay \$10 for an initial scan were willing to pay for an update of the card; 18.6% would pay \$5, 3.5% would pay another \$10.

Of the respondents who indicated that they have problems with the fit of apparel (Table C-5), 42.3% (n=81) were willing to pay \$5 for an initial body scan and 23.4% were willing to pay \$10 for an initial scan. More than half (51.4%) of the respondents with fit problems were willing to pay \$5 for an updated body scan. Only 4% were willing to pay \$10 for an update (Table C-6).

Accessibility

Accessibility of body scanning technologies was important with 51.5% of the respondents willing to have a scan if there was one per dressing room area. Approximately one-fourth (24.7%) would have a scan if there was only one per store; only 13.4% of respondents required body scan technologies in their dressing room before they would use it. Of the respondents, 41.2% would be more willing to have a body scan if they could wear a body suit during the process.

Research Objective 2

Current sizing problems with the fit of everyday apparel and jeans were measured. Specific fit problems relate to the waist, length and hips of jeans, the number of pairs tried on, and the alterations needed for the length, waist and inseam. In order to do this respondents were asked to reveal the size(s) and size category(ies) they wore.

Everyday apparel

Forty-four percent indicated that they sometimes alter their everyday apparel to fit. More than half (50.5%) would be willing to pay more for apparel made to their own size specifications. Some respondents (17.5%) have already purchased made-to-measure apparel.

Jeans

Many respondents have problems with the fit of jeans (mean 3.3/5 point Likert scale). Problems with the fit of the waist is experienced by 51.5% of respondents who wear jeans (n=84). Length is a problem for 49.5% of respondents. The hip area is a problem for 45.4% of respondents. The majority of respondents (75.3%) must try on several sizes when trying on different brands of jeans. Approximately half (45.4%) try on 1 to 3 pairs of jeans before they find a pair that fits. Some (19.6%) have to try on 4 to 6 pairs, while others (9.3%) try on 7 or more pairs of jeans before they find jeans that fits.

In addition to having problems finding jeans that fit, some people alter jeans. The length of the jeans is the most often altered part of jeans; 16.5% alter sometimes and 10.3% always alter the length of jeans. Sometimes respondents alter the waist (13.4%) of their jeans. The inseam is another area altered on jeans, but only 6.2% of respondents indicated that they sometimes and 1% always alters the inseam.

Testing of the Hypotheses

H1A: Age has no effect on the appeal of application (a).

A1A: Age has an effect on the appeal of application
(a).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were statistically significant. Results were not found to be significant for application (a) ($P=0.3808$) indicating that the age groups showed no difference in the appeal of application (a), made-to-measure (Appendix D, Table D-1). There is no evidence that age has an effect on the appeal of application (a).

H1B: Age has no effect on the appeal of application (b).

A1B: Age has an effect on the appeal of application
(b).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were statistically significant. Results were not found to be

significant for application (b) ($P=0.3453$) indicating that the age groups showed no difference in the appeal of application (b), data card (Appendix D, Table D-3). There is no evidence that age has effect on the appeal of application (b).

H1C: Age has no effect on the appeal of application (c).

A1C: Age has an effect on the appeal of application (c).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were statistically significant. Results were not found to be significant for application (c) ($P=0.5764$) indicating that the age groups showed no difference in the appeal of application (c), computer imaging (Appendix D, Table D-5). There is no evidence that age has effect on the appeal of application (c).

Results from the chi-square test indicated that consumers in all four age groups found either the data card or the computer imaging the most appealing body scan application; the chi-square statistic was not significant ($P=0.4170$) (Appendix C, Table C-7). Consumers below age 30 and between the ages of 40 and 49 thought the data card was most appealing while those age 30-39 found the computer imaging most appealing. The oldest groups (50+) found both the data card and computer imaging most appealing. There is no evidence of an association between age and appeal.

H2A: Age has no effect on the use of application (a).

A1A: Age has an effect on the use of application
(a).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were statistically significant. Results were not found to be significant for application (a) ($P=0.2341$) indicating that the age groups showed no difference in the use of application (a), made-to-measure (Appendix D, Table D-7). There is no evidence that age has an effect on the use of application (a).

H2B: Age has no effect on the use of application (b).

A1B: Age has an effect on the use of application
(b).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were statistically significant. Results were not found to be significant for application (b) ($P=0.3652$) indicating that the age groups showed no difference in the use of application (b), data card (Appendix D, Table D-9). There is no evidence that age has an effect on the use of application (b).

H2C: Age has no effect on the use of application (c).

A1C: Age has an effect on the use of application
(c).

Analysis of variance (ANOVA) was used to determine if any of the observed differences between the four age groups were

statistically significant. Results were not found to be significant for application (c) ($P=0.9186$) indicating that the age groups showed no difference in the use of application (c), computer imaging (Appendix D, Table D-11). There is no evidence that age has an effect on the use of application (c).

Results from the chi-square tests were used to compare the four age groups in terms of their use of body scanning applications (a), (b), and (c); the chi-square statistic was not significant ($P=0.5120$) (Appendix C, Table C-8). Three age groups of consumers (below 30, 40-49, and 50+) all claimed they would most use the data card. Only those consumers age 30-39 would most use computer imaging. There is no evidence of an association between age and use.

H3A: Body size has no effect on the appeal of application (a).

A3A: Body size has an effect on the appeal of application (a).

An ANOVA test determined if a statistically significant difference existed in the appeal of application (a) among the three body size groups. Results were not found to be significant ($P=0.8513$) for application (a), indicating that the three groups showed no difference in the appeal of application (a) (Appendix D, Table D-13). There is no evidence that body size has an effect on the appeal of application (a).

H3B: Body size has no effect on the appeal of application (b).

A3B: Body size has an effect on the appeal of application (b).

An ANOVA test determined if a statistically significant difference existed in the appeal of application (b) among the three body size groups. Results were not found to be significant ($P=0.9152$) for application (b), indicating that the three groups showed no difference in the appeal of application (b) (Appendix D, Table D-15). There is no evidence that body size has an effect on the appeal of application (b).

H3C: Body size has no effect on the appeal of application (c).

A3C: Body size has an effect on the appeal of application (c).

An ANOVA test was conducted to determine if any of the observed differences between the three body size groups were statistically significant. Results were not found to be significant for application (c) ($P=0.6502$) indicating that the age groups showed no difference in the appeal of application (c) (Appendix D, Table D-17). There is no evidence that body size has an effect on the appeal of application (c).

Chi-square tests revealed that the three body size groups found different applications most appealing; the chi-square statistic was not significant ($P=0.1130$) (Appendix C, Table C-9). The small size consumers found the made-to-measure and

data card (applications a & b) most appealing. Average size respondents thought the data card (application b) most appealing. The computer imaging (application c) was most appealing to the large consumers. There is no evidence of an association between body size and appeal. The appeal of the computer imaging may be a solution to the fit problems of large-size women reported by Chowdary and Beale (1988).

H4A: Body size has no effect on the use of application (a).

A4A: Body size has an effect on the use of application (a).

An ANOVA test determined if a statistically significant difference existed in the use of application (a) among the three body size groups. Results were not found to be significant ($P=0.7355$) for application (a), indicating that the three groups showed no difference in the use of application (a) (Appendix D, Table D-19). There is no evidence that body size has an effect on the use of application (a).

H4B: Body size has no effect on the use of application (b).

A4B: Body size has an effect on the use of application (b).

An ANOVA test determined if a statistically significant difference existed in the use of application (b) among the three body size groups. Results were not found to be significant ($P=0.6584$) for application (b), indicating that

the three groups showed no difference in the use of application (b) (Appendix D, Table D-21). There is no evidence that body size has an effect on the use of application (b).

H4C: Body size has no effect on the use of application (c).

A4C: Body size has an effect on the use of application (c).

An ANOVA test determined if a statistically significant difference existed in the use of application (c) among the three body size groups. Results were not found to be significant ($P=0.3292$) for application (c), indicating that the three groups showed no difference in the frequency of use of application (c) (Appendix D, Table D-23). There is no evidence that body scanning has an effect on the use of application (c).

Chi-square tests revealed that the three body size groups would most often use the same applications that they found most appealing. The chi-square statistic was significant ($P=0.0260$) (Appendix C, Table C-10). A greater number of small-size consumers than expected would most often use the made-to-measure application (application a). In addition, more large-size consumers than expected would most often use the computer imaging (application c). There is evidence of an association between body size and the most often used application. DeLong, Ashdown, Butterfield and Turnbladh

(1990) reported that those women who have fit problems have to look to custom fit apparel. This finding that there is an association between body size and most often use offers small, average and large-size women alternatives to custom fit apparel.

H5: There is no association between amount spent on wardrobe and fit problems.

A5: There is an association between amount spent on wardrobe and fit problems.

Chi-square tests revealed that those two groups of respondents (\$200 or less, \$1,000 or more) reported less problems with fit than the remaining two groups (spending \$200-499, 500-999); the chi-square statistic was not significant ($P=0.5140$). There is no evidence of an association between amount spent on wardrobe and fit problems.

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

The purpose of this study was to determine the feasibility of a body scanning system for consumer use at retail. The Engel, Blackwell, and Miniard Decision Process Model (1993) was the conceptual framework used in this study and this study focused on the first stage, need recognition. Need recognition essentially depends on how much discrepancy exists between the actual state (i.e., the consumer's current situation) and the desired state (i.e., the desired situation). When this discrepancy exceeds a certain level, a need is recognized.

Data was collected using a five-page questionnaire mailed to the sample in February, 1994. The questionnaire consisted of items relating to the acceptability of body scanning, sizing, fit problems, as well as demographics. The sample chosen for the study consisted of 200 female SPA employees at the University of North Carolina at Greensboro; a total of 97 (48.5% response rate) returned the questionnaire.

The women in the sample were employed in positions ranging from research assistant to secretarial to housekeeping. Two-fifths of the sample had an annual household income of the sample of \$50,000 and above. The

females in the study were ages 18 and older, but the largest portion was 30-39. The sample was almost all white with some black respondents. The education of the sample ranged from holding a high school diploma to graduate degrees. The majority of the women in the study spent between \$200 and \$1,000 on their wardrobe last year. Nearly all the respondents had used a computer at work.

Research Objective 1

Consumers interest in body scanning was examined, specifically, acceptability of the three body scan outputs: made-to-measure, data card, and computer imaging. The respondents were willing to have a body scan. All three of the body scan applications were found to be appealing. The data card was the most appealing followed by the computer imaging. The made-to-measure application was the least appealing application of the three. Some respondents reported very strong feelings about the applications and stated that they thought all the applications were very appealing. Other respondents possessed very strong feelings that none of the applications were appealing.

In general, consumers' response to each of the three body scan applications was examined. For each of the three applications about half of the respondents (50-54.7%) rated the applications equally appealing and usable. Other respondents (36.5-44.8%) found the applications more appealing

than usable. Only a small percentage of respondents (5.2-9.3%) found the applications more usable than appealing.

Consumers' usage of the three body scan applications was similar to the appeal. The data card would be most used followed by the computer imaging and made-to-measure. A portion of the respondents indicated that they would not use any of the applications. These consumers can be characterized as having very few fit problems with no need to try on several sizes. The appeal and usage of the three applications was consistent for the data card and computer imaging but not for the made-to-measure. Even though some respondents found it appealing, they would not necessarily use it.

Consumers would use the body scan technologies when purchasing bottoms (jeans and slacks), swimwear, and tops (blouses and shirts). They were willing to pay for initial scans as well as for updates (range \$5-\$10). The results indicate that the body scan technology needs to be located only one per dressing room area. Less than half of the respondents would be more willing to have a scan if they could wear a bodysuit.

Research Objective 2

Current sizing problems with the fit of everyday apparel and jeans were measured. Almost half of the respondents indicated that they sometimes alter their everyday apparel. More than half would be willing to pay more for apparel made

to their own size specifications. Fit problems with jeans occur in the waist, hip area, and length. It is common for respondents to have to try on more than one pair to find proper fit.

Testing of the Hypotheses

Five research hypotheses were developed to answer research objectives 3 (Hypotheses 1-4) and 4 (Hypothesis 5). Means and frequencies were run to provide descriptive statistics. Analysis of variance (ANOVA) tests were used to test Hypotheses 1-4. A chi-square test was used to test Hypothesis 5.

ANOVA tests were used to test Hypotheses 1-4. None of the hypotheses tested produced significant results at the .05 level. Results indicated that the age of the consumer does not effect the appeal or usage of the three body scanning applications. In addition, consumers' body size has no effect on the appeal or usage of the three body scanning applications. The mean scores for Hypotheses 1-4 were examined; no clear pattern emerged. For example, the younger consumers and large body size consumers did not always have the highest mean scores on appeal and usage.

Chi-square tests revealed an association between body size and the most often used body scan application; the chi-square statistic was significant ($P=0.0260$). A greater number of small-size respondents than expected would most often use

the made-to-measure application. More large-size consumers than expected would most use the computer imaging application.

Chi-square tests were used to test Hypothesis 5; the results were not statistically significant. There is no evidence of an association between amount spent on wardrobe and fit problems.

Conclusions

This study was designed to determine consumers' interest in body scanning technologies. The need recognition stage of the EBM Decision Process Model (Engel, Blackwell, & Miniard, 1993) was found to be useful in providing an integrated approach to this assessment. Body scanning technologies are feasible for the apparel market with employed female consumers.

Results indicated that the majority (76%) of the sample were willing to have a body scan. Neither age nor body size had an effect on the appeal or use of the three applications. This may indicate that for this sample feasibility is not an age or body size issue.

A portion of the respondents (16.5%) found all three body scanning applications very appealing. These consumers were characterized as having fit problems with tops and bottoms and having to try on several size garments.

A part of the sample (n=11, 11.3%) did not find any of the applications appealing and 10 (of 11) of these consumers

claimed that they would not use any of the three applications. These two groups of respondents are the same consumers who experience very few fit problems and do not try on several sizes before finding the correct fit.

The respondents were asked to indicate how appealing and potentially usable each of the three body scan applications were to them. Even though the three applications were found appealing and usable, in general responses appeared much more appealing than usable. Further examination revealed that for each of the three applications: (1) some respondents (36.5-44.8%) found the applications more appealing than usable, (2) almost half (50.0-54.2%) found the applications equally appealing and usable, and (3) a few respondents (5.2-9.3%) found the applications more usable than appealing.

The data card was both most appealing and most usable followed by the computer imaging and the made-to-measure. The made-to-measure application, the preferred application of manufacturers, was not as appealing as the data card or computer imaging.

Consumers would use body scanning before purchasing jeans and slacks, swimwear, and blouses and shirts. Not only did consumers indicate that they would be willing to have a body scan, but they were also willing to pay for \$5 (36.1%) and \$10 (20.6%) for an initial scan. There were even consumers willing to pay \$5 (44.3%) and \$10 (3.1%) each time they updated their card.

Implications

The following are implications for educators. This study:

- 1) reinforces the use of the conceptual framework (EBM) to identify need recognition, the first stage, and
- 2) contributes to the void in the literature pertaining to the market feasibility of body scanning and size prediction technologies at retail.

Implications for retailers and manufacturers from results of this study are to:

- 1) consider body scanning technologies, especially the data card and computer imaging, and
- 2) adopt body scanning for bottoms, swimwear and tops.

Recommendations

Recommendations for retailers and manufacturers from results of this study are to:

- 1) further explore consumers' appeal and usage of body scanning technologies by pre- and post-testing actual consumer exposure to the technologies, and
- 2) consider the purchase of body scanning technologies for in-store use.

Recommendations for future study are to:

- 1) further explore demographic variables that may give insight on the consumer's appeal and use of the three applications, such as race and income,

- 2) select a niche market such as athletes or fitness market to compare fit problems, and
- 3) add additional variables to the instrument to obtain even more specific fit and sizing problems.

REFERENCES

- American Psychological Association. (1992). Publication manual of the American Psychological Association (3rd ed.). Washington, D.C.: Author.
- Brown, P. (1992). Ready-to-wear apparel analysis. New York: Macmillan Publishing Co.
- C-Stores coming around to POS scanning. (1994, January). Retail Information Systems News, pp.6-7.
- Cassill, N., & Drake, M.F. (1987). Apparel selection criteria related to female consumers' lifestyle. Clothing and Textiles Research Journal, 6(1), 20-28.
- Chandler, B. (1992, October). Multimedia: Multifaceted retail tool. Discount Merchandiser, p.39.
- Chowdary, U., & Beale, N. (1988). Plus-size women's clothing interest, satisfactions and dissatisfactions with ready-to-wear apparel. Perceptual and Motor Skills, 66, 783-788.
- DeLong, M., Ashdown, S., Butterfield, L., & Turnbladh, K.F. (1993). Data specification needed for apparel production using computers. Clothing and Textiles Research Journal, 11(3), 1-7.
- DeLong, M., LaBat, K., & Bye, E. (1991). University/industry collaborative research: Powerful partners. Journal of Home Economics, Summer, 7-13.
- Deonier, C., DeLong, M., & Martin, F. (1979). Weight loss and resulting fit and size change of ready-to-wear for American women. Home Economics Research Journal, 7, 186-205.
- DuCoffe, J., & Cohen, S. (1980). Making it big. New York: Simon & Schuster.
- Engel, J.F., Blackwell, R.D., & Miniard, P.W. (1993). Consumer behavior (7th ed.). Chicago, IL: The Dryden Press.
- Gray, S. (1994, January). Formula for a fashion show. Bobbin, pp.54-58.

- Grill, D.S., & Sharkley, M.F. (1991). Fairchild fact file - the textile/apparel industries. New York: Capital Cities Media.
- Harris, M., Mehrman, M., & Dougherty, J. (1992, March). Made-to-measure: Computing a great fit. Bobbin, pp. 56-61.
- International Labor Office. (1992). Year book of labor statistics. (51st ed.). Geneva, Switzerland: Author.
- Jacobs, B. (1994, January). Operating stores: High octane execution. Chain Store Age Executive, pp. 21-24.
- Jarnow, J., & Guerreiro, M. (1991). Inside the fashion business (5th ed.). New York: Macmillan Publishing Co.
- Kerin, R., Jain, A., & Howard, D. (1992). Store shopping experience and consumer price-quality-value perceptions. Journal of Retailing, 68, 376-397.
- LaBat, K., & DeLong, M. (1990). Body cathexis and satisfaction with fit of apparel. Clothing and Textiles Research Journal, 8(2), 43-47.
- Martell, C.R. (1990, January). Three-dimensional thinking. Apparel Manufacturer, pp. 8-16.
- Morganstein, M., & Strongin, H. (1987). Modern Retailing. New York: John Wiley & Sons, Inc.
- O'Hare, W. (1993, November). Top towns for working women. American Demographics, pp. 44-47.
- Oliver, B.A., Bickle, M.C., & Shim, S. (1993). Profile of made-to-measure customers: Body characteristics and purchase selection. Clothing and Textiles Research Journal, 11(2), 59-62.
- Person, J.E. (Ed.). (1993). Statistical Forecast of the United States. Detroit, MI: Gale Research Inc.
- Quick response technologies. (1991, March). Chain Store Age Executive, pp. 6-7.
- Robins, G. (1992, June). Auto ID update. Stores, pp 31-33.
- Rowland, R. (1990, August). Partners in retail technology: Marketing and electronics. Discount Merchandiser, pp. 85-89.

- Shim, S., & Drake, M.F. (1988). Apparel selection by employed women: A typology of information search patterns. Clothing and Textiles Research Journal, 6(2), 1-9.
- Shim, S., & Drake, M.F. (1989). Information search in the purchase of employment apparel: A synthesis of two theories. Clothing and Textiles Research Journal, 7(3), 40-46.
- Shim, S., & Kotsiopulos, A. (1990). Women's physical size, body-cathexis, and shopping for apparel. Perceptual and Motor Skills, 61, 1031-1042.
- Shim, S., & Kotsiopulos, A. (1991). Big and tall men as apparel shoppers: Consumer characteristics and shopping behavior. Clothing and Textiles Research Journal, 9(2), 16-24.
- Shim, S., Kotsiopulos, A., & Knoll, D. (1991). Short, average-height, tall, and big men: Body-cathexis, clothing and retail satisfactions, and clothing behavior. Perceptual and Motor Skills, 70, 83-96.
- Shortridge, B.G. (1987). Atlas of American Women. New York: Macmillan Publishing Co.
- Solomon, M. (1985). The Psychology of Fashion. U.S.: D.C. Heath & Co.
- Tamburrino, N. (1992, May). Apparel sizing issues, part 2. Bobbin, pp. 52-60.
- Tamburrino, N. (1992, June). Sized to Sell. Bobbin, pp. 68-74.
- Textile/Clothing Technology Corporation demonstrated new apparel manufacturing technology at recent apparel research conference. (1993, January). Textile News, p. 8.
- Touch screen technology now delivers quick R.O.I. (1994, January). Chain Store Age Executive, pp.1-4.
- U.S. Department of Commerce. (1993). Survey of current business, 73(9), Washington, DC: Author.
- Waldrop, J., & Mogelonsky, M. (1992). The Seasons of Business. U.S.: American Demographics Books.

Wallach, J. (1986). Big world of petites. Stores, 68.

Woolf, H. (Ed.). (1976). Webster's New Collegiate Dictionary. Springfield, MA: G & C Merriam Co.

APPENDIX A
COVER LETTER, QUESTIONNAIRE, AND
HUMAN SUBJECTS APPROVAL FORM

THE UNIVERSITY OF NORTH CAROLINA
GREENSBORO

School of Human Environmental Sciences
Department of Clothing and Textiles

February 10, 1994

1 ~

Dear 2 ~ :

Have you ever been dissatisfied with the fit of your clothes? We all would like to have clothes made for our specific body measurements. The Clothing and Textiles Department at UNCG is interested in helping consumers and manufacturers understand the nature of apparel "fit" problems. This research focuses on the use of body scanning technology to help consumers purchase better fitting clothes.

You are part of a carefully selected sample of female consumers. I would greatly appreciate it if you would complete the enclosed questionnaire and return it through campus mail. A mailing label has been attached so that all you have to do is fold and staple the questionnaire. The survey will take approximately 10 minutes to complete.

You are assured of complete confidentiality. The questionnaire has an identification number for mailing purposes only. The number enables us to remove your name from the mailing list when the questionnaire is returned. Your name will never be placed on the questionnaire.

Please return the completed survey by **February 28, 1994**. Thank you in advance for your time and interest.

Sincerely,

Audra Knight
Graduate Student

Nancy L. Cassill, Ph.D.
Associate Professor
Thesis Advisor

Enclosure

SECTION I

DIRECTIONS: Please read the description of a body scan experience and answer the related questions below.

Body scanning technologies are being developed to take accurate measurements of the human body in order to provide better fitting apparel to the consumer. A typical body scan experience includes visiting your favorite retail store where body scan booths are located. The specially designed computerized booth can be found in the dressing room area of the store. The inside of the booth is black with dim lighting and is operated solely by the consumer. The booth is larger than the average dressing room with ample room to remove all articles of clothing, except your underwear, before the body scan. After you have removed your clothes and are standing at the specified location in the booth, you simply push a button and your measurements will be taken.

The booth has a non-contact data gathering device consisting of a series of video cameras connected to a computer where the data is used to make size predictions. No video image of your body is produced and the actual measurement time is only 6 seconds. When the measurement process is over you may get dressed and leave the booth. At this point you can choose to have your measurements transferred to an electronic card or kept in a database or both. If you choose the card, it will be generated within a few minutes and you can take it with you. If you choose to have your measurements stored on the database you can access them whenever you visit a retail store with this technology. The body measurement information can be updated as often as you like.

1. Would you be willing to have a body scan?
☐ yes
☐ no
2. How much would you be willing to pay for a body scan? Assume the cost would include the data card and/or data storage and retrieval at the retail store.

<u>Initial Time</u>	<u>Update</u>
☐ \$ 0.00	☐ \$ 0.00
☐ \$ 5.00	☐ \$ 5.00
☐ \$10.00	☐ \$10.00
3. How accessible does a body scanner have to be in order for you to use it?
☐ one per store
☐ one per dressing room area
☐ every dressing room
4. Would you be more willing to have a body scan if you could wear a body suit during the process?
☐ yes
☐ no
5. Do you think it is worth your time to have a body scan in order to have better fitting apparel?
☐ yes
☐ no

SECTION II

DIRECTIONS: Please read the brief passages below on the possible applications of body scanning for apparel purchases and answer the related questions.

- (a) Body scanning can produce made-to-measure clothing. It will be possible to have your own measurements, stored on a card or in a database, transmitted directly to the manufacturer so that they can produce apparel to meet your size specifications. All you have to do is visit your favorite retail store, have your body scanned, select an item of apparel you would like to purchase and choose a style and color. Once you have decided on a style and color, that information along with your measurements will be transmitted to the manufacturer where your apparel will be cut and sewn to your size specifications. Your clothes will arrive at your home in about 5 days.
- (b) Body scanning can generate a data card with your body measurements. This card can be used three ways:
 - 1) When you want to make a direct mail (catalog) purchase, you will insert the data card in the slot on the phone. As the operator accesses your body dimensions, the correct garment size for that particular item is sent directly to you.
 - 2) When you want to make a "convenience" purchase in a retail store, you will insert the card in a machine in the retail store and the correct size for a particular brand will be selected for you. This may eliminate your trying on many different garments thus saving you time and money.
 - 3) Your card can be sent to someone who wishes to purchase a gift for you. For example, if the card is sent to a grandparent, the grandparent can come into the retail store, "call up" your body dimensions, and purchase a product for you that "fits."
- (c) Body scanning can project your image "on screen" with a particular product superimposed on your body. This will enable you to look at a computer screen and "try-on" clothes on-screen. Instead of carrying 10 items into the dressing room, you may "try-on" the apparel on-screen first to see how they will look on your computer image. This will enable you to automatically rule out clothes that do not fit your body well.

		Not at all		Somewhat		Very
6.	Is application (a) appealing to you?	1	2	3	4	5
7.	Is application (b) appealing to you?	1	2	3	4	5
8.	Is application (c) appealing to you?	1	2	3	4	5
9.	Which application is most appealing to you? (Circle one) (a) (b) (c)					

		Not at all		Somewhat		Very
10.	How often would you use application (a)?	1	2	3	4	5
11.	How often would you use application (b)?	1	2	3	4	5
12.	How often would you use application (c)?	1	2	3	4	5
13.	Which application would you use most frequently? (Circle one) (a) (b) (c)					
14.	For what apparel products would you be most likely to use body scanning? (Circle all that apply)					
	Hosiery	Jeans/slacks	Swimsuits	Sleepwear	Undergarments	
	Blouses/shirts	Jackets	Exercise Apparel	Other _____		

SECTION III

DIRECTIONS: Please answer the following questions related to the fit of garments.

15. Do you currently have problems with the fit of any garments?
 ___ yes-----> ___ tops
 ___ no ___ bottoms
 ___ both
16. Do you have to try on several size garments before you find one that fits?
 ___ yes
 ___ no
17. What size do you most often purchase in shirts/blouses? Put the number of the size in the blank. (You may fill in more than one blank.)
 Junior _____
 Junior Petite _____
 Missy _____
 Missy Petite _____
 Womens _____
18. What size do you most often purchase in jeans/slacks? Put the number of the size in the blank. (You may fill in more than one blank.)
 Junior _____
 Junior Petite _____
 Missy _____
 Missy Petite _____
 Womens _____

19. How often do you have your everyday apparel altered to fit?
☐ always
☐ sometimes
☐ never
20. Would you be willing to pay more for everyday apparel made to your own size specifications?
☐ yes-----> How much more? \$ _____
☐ no
21. Have you ever purchased made-to-measure clothing?
☐ yes-----> What was it? _____
☐ no
22. Do you wear jeans?
☐ yes
☐ no (skip to Question 27)
23. Do you currently have problems with the fit of jeans? (Check all that apply)
- | | | |
|------------|--------------------------|--------------------------|
| | yes | no |
| Length --> | ☐ | ☐ |
| Waist ---> | ☐ | ☐ |
| Hips ----> | ☐ | ☐ |
24. When trying on several brands of jeans do you have to try on several sizes before you find one that fits?
☐ yes -----> Average number tried on (Circle) 1-3 4-6 7-9 10+
☐ no
25. How difficult is it for you to find a pair of jeans that fits?
- | | | | | |
|------|---|---|---|-----------|
| 1 | 2 | 3 | 4 | 5 |
| Easy | | | | Difficult |
26. How often do you have your jeans altered to fit?
- | | | | |
|---------------|-------|---|--------|
| | Never | | Always |
| Length -----> | 1 | 2 | 3 |
| Waist -----> | 1 | 2 | 3 |
| Inseam -----> | 1 | 2 | 3 |

SECTION IV

DIRECTIONS: Please circle the correct answer.

27. What is your age?
1. Below 25
 2. 25-29
 3. 30-39
 4. 40-49
 5. 50-59
 6. 60+
28. What is your household income level?
1. Below \$15,000
 2. \$15,000-19,999
 3. \$20,000-29,999
 4. \$30,000-49,999
 5. \$50,000-69,999
 6. \$70,000 or above
29. What is your ethnic background?
1. Asian
 2. Black
 3. Hispanic
 4. White
30. What is the highest level of education you have completed?
1. Some high school
 2. High school diploma
 3. Some college or vocational training beyond high school
 4. Bachelor's degree
 5. Some graduate school
 6. Graduate degree
31. How much did you spend on your wardrobe last year?
1. Less than \$200
 2. \$200-499
 3. \$500-999
 4. \$1,000+
32. Do you use a computer?
- ____ yes
- ____ no
33. Is there anything you'd like to tell us about the size and/or fit of apparel?

**THANK YOU FOR YOUR TIME - PLEASE FOLD AND
STAPLE THE QUESTIONNAIRE BEFORE DROPPING
IT IN CAMPUS MAIL.**

934108

UNIVERSITY OF NORTH CAROLINA AT GREENSBORO

Institutional Review Board
Notification Form

DATE: January 31, 1994
PROJECT TITLE: The Market Feasibility of Body Scanning
& Size Prediction Technologies at Retail
PRINCIPAL INVESTIGATOR: Nancy Cassell
SCHOOL/COLLEGE: HES DEPARTMENT: CTX

ACTION TAKEN:

☒ Exempt
☐ Expedited Review
☐ Full IRB Review

DISPOSITION OF APPLICATION:

☐ Approved
☐ Disapproved

MODIFICATIONS AND COMMENTS:

Annelys E. Maddox-Britt
IRB Chair/Designee

Approval of research is valid for one year. If your research goes beyond one year, the project must be reviewed prior to continuation.

APPENDIX B
SAMPLE CHARACTERISTICS

Table B-1

Description of Respondents

Characteristic	Frequency	Percentage
<u>Willing to have a body scan?</u>		
Yes	23	23.7
No	74	76.3
<u>How much for initial scan?</u>		
\$0.00	35	36.1
\$5.00	35	36.1
\$10.00	20	20.6
No answer	7	
<u>How much for update?</u>		
\$0.00	42	43.2
\$5.00	43	44.3
\$10.00	3	3.1
No answer	9	
<u>How accessible does scanner have to be?</u>		
One per store	24	24.7
One per dressing room area	50	51.5
One per dressing room	13	13.4
No answer	10	
<u>More willing if wear bodysuit</u>		
Yes	40	41.2
No	55	56.7
No answer	2	
<u>Worth time to have a scan?</u>		
Yes	65	67.0
No	31	32.0
No answer	1	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Appeal of Application (a)</u>		
(1) Not at all	20	20.6
(2)	5	5.2
(3) Somewhat	27	27.8
(4)	12	12.4
(5) Very	33	34.0
<u>Appeal of Application (b)</u>		
(1) Not at all	13	13.4
(2)	7	7.2
(3) Somewhat	19	19.6
(4)	24	24.7
(5) Very	33	34.0
No answer	1	
<u>Appeal of Application (c)</u>		
(1) Not at all	15	15.5
(2)	11	11.3
(3) Somewhat	17	17.5
(4)	18	18.6
(5) Very	35	36.0
No answer	1	
<u>Application Most Appealing</u>		
(a) Made-to-measure	21	21.6
(b) Data card	32	33.0
(c) Computer imaging	31	32.0
No answer	13	
<u>Use of Application (a)</u>		
(1) Not at all	20	20.6
(2)	21	21.6
(3) Somewhat	29	29.9
(4)	15	15.5
(5) Very often	11	11.3
No answer	1	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Use of Application (b)</u>		
(1) Not at all	13	13.4
(2)	18	18.6
(3) Somewhat	17	17.5
(4)	28	28.9
(5) Very often	20	20.6
No answer	1	
<u>Use of Application (c)</u>		
(1) Not at all	18	18.6
(2)	16	16.5
(3) Somewhat	20	20.6
(4)	21	21.6
(5) Very often	21	21.6
No answer	1	
<u>Application Most Used</u>		
(a)	18	18.6
(b)	36	37.1
(c)	31	32.0
No answer	12	
<u>Would use body scanning for:</u>		
Hosiery	5	5.2
Jeans/slacks	73	75.3
Swimsuits	52	53.6
Sleepwear	5	5.2
Underwear	32	33.0
Blouses/shirts	51	52.6
Jackets	46	47.4
Exercise apparel	12	12.4
Skirts	7	7.2
Shoes	4	4.1
Dresses	18	18.6
Suits	6	6.2
Coats	3	3.1
<u>Fit problems</u>		
Yes	81	83.5
No	14	14.4
No answer	2	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Problems with tops</u>		
Yes	47	48.5
No	34	35.1
No answer	16	
<u>Problems with bottoms</u>		
Yes	73	75.3
No	8	8.2
No answer	16	
<u>Problems with tops and bottoms</u>		
Yes	40	41.2
No	41	42.3
No answer	16	
<u>Try several garments for fit</u>		
Yes	74	76.3
No	21	21.6
No answer	2	
<u>Junior Sizes - Tops</u>		
5	1	1.0
6	2	2.1
7	1	1.0
9	3	3.1
10	2	2.1
11	1	1.0
12	1	1.0
13	2	2.1
14	1	1.0
No answer	83	
<u>Junior Petite Sizes - Tops</u>		
6	1	1.0
8	1	1.0
No answer	95	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Missy Sizes - Tops</u>		
3	1	1.0
5	1	1.0
6	6	6.2
7	2	2.1
8	6	6.2
9	4	4.1
10	18	18.6
11	2	2.1
12	9	9.3
13	3	3.1
14	8	8.2
16	3	3.1
No answer	34	
<u>Missy Petite Sizes - Tops</u>		
3	1	1.0
4	2	2.1
6	4	4.1
8	2	2.1
10	1	1.0
12	1	1.0
13	1	1.0
14	1	1.0
No answer	84	
<u>Womens Sizes - Tops</u>		
10	1	1.0
12	1	1.0
14	2	2.1
15	1	1.0
16	4	4.1
18	2	2.1
20	2	2.1
22	1	1.0
38	1	1.0
No answer	82	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Junior Sizes - Bottoms</u>		
2	1	1.0
8	2	2.1
9	3	3.1
10	3	3.1
11	1	1.0
15	1	1.0
No answer	86	
<u>Junior Petite Sizes - Bottoms</u>		
2	1	1.0
8	1	1.0
9	1	1.0
14	1	1.0
No answer	93	
<u>Missy Sizes - Bottoms</u>		
3	1	1.0
5	1	1.0
6	2	2.1
7	3	3.1
8	9	9.3
9	3	3.1
10	8	8.2
11	2	2.1
12	11	11.3
13	3	3.1
14	10	10.3
15	3	3.1
16	2	2.1
18	1	1.0
No answer	38	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Missy Petite Sizes - Bottoms</u>		
3	1	1.0
4	3	3.1
6	2	2.1
8	2	2.1
9	1	1.0
10	1	1.0
11	1	1.0
12	2	2.1
13	1	1.0
14	2	2.1
16	1	1.0
No answer	80	
<u>Womens Sizes - Bottoms</u>		
10	1	1.0
12	2	2.1
14	3	3.1
15	1	1.0
16	3	3.1
18	2	2.1
20	2	2.1
22	2	2.1
40	1	1.0
No answer	80	
<u>Everyday apparel altered to fit</u>		
Always	4	4.1
Sometimes	43	44.3
Never	47	48.5
No answer	3	
<u>Pay more for apparel made for you</u>		
Yes	49	50.5
No	43	44.3
No answer	5	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>How much more?</u>		
\$5	6	6.2
\$7	9	9.3
\$8	1	1.0
\$10	9	9.3
\$12	1	1.0
\$15	5	5.2
\$20	2	2.1
\$35	1	1.0
\$40	1	1.0
No answer	62	
<u>Purchased made-to-measure</u>		
Yes	17	17.5
No	78	80.4
No answer	2	
<u>Made-to-measure was?</u>		
Wedding gown	4	4.1
Swimsuit	1	1.0
Dress	7	7.2
Skirt	1	1.0
Bridesmaid dress	1	1.0
Evening gown	1	1.0
Suit	2	2.1
No answer	80	
<u>Wear jeans</u>		
Yes	84	86.6
No	10	10.3
No answer	3	
<u>Problems with length of jeans</u>		
Yes	48	37.1
No	36	49.5
No answer	13	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Problems with the waist of jeans</u>		
Yes	50	51.5
No	33	34.0
No answer	14	
<u>Problems with the hips of jeans</u>		
Yes	44	40.2
No	39	45.4
No answer	14	
<u>Try on several sizes of jeans</u>		
Yes	73	75.3
No	12	12.4
No answer	12	
<u>Number try on</u>		
1-3	46	47.4
4-6	19	19.6
7-9	4	4.1
10+	5	5.2
	22	
<u>Difficulty to find jeans that fit</u>		
(1) Easy	5	5.2
(2)	14	14.4
(3)	28	28.9
(4)	21	21.6
(5) Difficult	16	16.5
No answer	13	
<u>Alter length of jeans</u>		
(1) Never	58	59.8
(2)	16	16.5
(3) Always	10	10.3
No answer	13	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Alter waist of jeans</u>		
(1) Never	71	73.2
(2)	13	13.4
(3) Always	0	0.0
No answer	13	
<u>Alter inseam of jeans</u>		
(1) Never	77	79.4
(2)	6	6.2
(3) Always	1	1.0
No answer	13	
<u>Age</u>		
Below 25	3	3.1
25-29	13	13.4
30-39	35	36.1
40-49	25	25.8
50-59	15	15.5
60+	4	4.1
No answer	2	
<u>Income</u>		
Below \$15,000	1	1.0
\$15,000-19,999	13	13.4
\$20,000-29,999	10	10.3
\$30,000-49,999	30	30.9
\$50,000-69,999	23	23.7
\$70,000 or above	16	16.5
No answer	4	
<u>Race</u>		
Asian	0	0.0
Black	11	11.3
Hispanic	1	1.0
White	82	84.5
No answer	3	

Description of Respondents (Cont'd)

Characteristic	Frequency	Percentage
<u>Education</u>		
Some high school	0	0.0
High school diploma	5	5.2
Some college or vocational training	41	42.3
Bachelor's degree	25	25.8
Some graduate school	10	10.3
Graduate degree	14	14.4
No answer	2	
<u>Amount spent on wardrobe</u>		
Less than \$200	10	10.3
\$200-499	44	45.4
\$500-999	29	29.9
\$1,000+	11	11.3
No answer	3	
<u>Use a computer at work</u>		
Yes	94	97.0
No	2	2.1
No answer	1	

Table B-2

Description of Respondents (Means)

Variable	N	Mean	Std. Dev.	Min.	Max.
Appeal of (a)	97	3.34	1.51	1	5
Appeal of (b)	96	3.59	1.38	1	5
Appeal of (c)	96	3.49	1.47	1	5
Use of (a)	96	2.75	1.27	1	5
Use of (b)	96	3.25	1.35	1	5
Use of (c)	96	3.11	1.42	1	5
Frequency of jeans alteration	94	1.53	.60	0	3
Number of jeans try on	75	1.55	.89	0	4
Difficulty to find jeans that fit	85	3.30	1.20	0	5
Frequency of altering jeans length	85	1.41	.71	0	3
Frequency of altering jeans waist	85	1.14	.38	0	2
Frequency of altering jeans inseam	85	1.08	.35	0	3

Table B-3

Vector Analysis - Appeal of Body Scan Applications

Appeal	Frequency	Percent
1 1 1	11	11.3
1 1 2	1	1.0
1 2 2	2	2.1
1 2 5	1	1.0
1 3 5	1	1.0
1 4 4	1	1.0
1 5 1	1	1.0
1 5 3	1	1.0
1 5 5	1	1.0
2 2 2	2	2.1
2 4 4	1	1.0
2 4 5	1	1.0
2 5 5	1	1.0
3 1 5	1	1.0
3 2 1	1	1.0
3 2 4	1	1.0
3 3 3	9	9.3
3 3 4	2	2.1
3 3 5	1	1.0
3 4 2	1	1.0
3 4 3	2	2.1
3 4 4	2	2.1
3 4 5	2	2.1
3 5 3	1	1.0
3 5 4	2	2.1
3 5 5	2	2.1
4 3 2	2	2.1
4 3 4	1	1.0
4 4 4	3	3.1
4 4 5	2	2.1
4 5 1	1	1.0
4 5 3	1	1.0
4 5 4	1	1.0
4 5 5	1	1.0
5 . .	1	1.0
5 3 2	1	1.0
5 3 5	2	2.1
5 4 2	2	2.1

Table B-3 (Cont'd)

Vector Analysis - Appeal of Body Scan Applications

Appeal	Frequency	Percent
5 4 4	4	4.1
5 4 5	3	3.1
5 5 1	1	1.0
5 5 3	3	3.1
5 5 5	16	16.5

Table B-4

Vector Analysis - Use of Body Scan Applications

Use	Frequency	Percent
. . .	1	1.0
1 1 1	10	10.3
1 1 2	1	1.0
1 2 2	1	1.0
1 2 5	2	2.1
1 3 3	1	1.0
1 3 5	2	2.1
1 4 1	1	1.0
1 4 2	1	1.0
1 4 4	1	1.0
2 2 2	6	6.2
2 2 3	2	2.1
2 3 1	1	1.0
2 3 3	3	3.1
2 4 2	1	1.0
2 4 3	1	1.0
2 4 4	2	2.1
2 5 3	2	2.1
2 5 4	2	2.1
2 5 5	1	1.0
3 1 4	1	1.0
3 2 1	2	2.1
3 2 2	2	2.1
3 2 3	1	1.0
3 2 4	1	1.0
3 2 5	1	1.0
3 3 2	1	1.0
3 3 3	1	1.0
3 3 4	2	2.1
3 3 5	1	1.0
3 4 1	2	2.1
3 4 4	2	2.1
3 4 5	6	6.2
3 5 1	1	1.0
3 5 3	2	2.1
3 5 4	1	1.0
3 5 5	2	2.1
4 3 2	2	2.1

Table B-4 (Cont'd)

Vector Analysis - Use of Body Scan Applications

Use	Frequency	Percent
4 3 3	1	1.0
4 3 4	1	1.0
4 4 3	1	1.0
4 4 4	4	4.1
4 4 5	2	2.1
4 5 3	2	2.1
4 5 4	2	2.1
5 1 3	1	1.0
5 3 2	1	1.0
5 4 1	1	1.0
5 4 3	1	1.0
5 4 5	2	2.1
5 5 3	1	1.0
5 5 4	2	2.1
5 5 5	2	2.1

APPENDIX C
CHI-SQUARE ANALYSES

Table C-1

Chi-Square Test - Appeal of Application (a) by
Use of Application (a)

APPEAL OF (a)	USE OF (a)					
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	1	2	3	4	5	Total
1	16	2	2	0	0	20
	4.17	4.38	6.04	3.13	2.29	
	33.61	1.29	2.70	3.13	2.29	
	16.67	2.08	2.08	0.00	0.00	20.83
	80.00	10.00	10.00	0.00	0.00	
	80.00	9.52	6.90	0.00	0.00	
2	2	3	0	0	0	5
	1.04	1.09	1.51	0.78	0.57	
	0.88	3.32	1.51	0.78	0.57	
	2.08	3.13	0.00	0.00	0.00	5.21
	40.00	60.00	0.00	0.00	0.00	
	10.00	14.29	0.00	0.00	0.00	
3	2	12	12	1	0	27
	5.62	5.91	8.16	4.22	3.09	
	2.34	6.29	1.81	2.46	3.09	
	2.08	12.50	12.50	1.04	0.00	28.13
	7.41	44.44	44.44	3.70	0.00	
	10.00	57.14	41.38	6.67	0.00	
4	0	1	5	6	0	12
	2.50	2.63	3.63	1.88	1.38	
	2.50	1.01	0.52	9.08	1.38	
	0.00	1.04	5.21	6.25	0.00	12.50
	0.00	8.33	41.67	50.00	0.00	
	0.00	4.76	17.24	10.00	0.00	
5	0	3	10	8	11	32
	6.67	7	9.67	5	3.67	
	6.67	2.29	0.01	1.8	14.67	
	0.00	3.13	10.42	8.33	11.46	33.33
	0.00	9.38	31.25	25.00	34.38	
	0.00	14.29	34.48	53.33	100.0	
Total	20	21	29	15	11	96
	20.83	21.88	30.21	15.63	11.46	100.0

Frequency Missing = 1

Chi-Square Statistic P=.0000

Table C-2

Chi-Square Test - Appeal of Application (b) by
Use of Application (b)

APPEAL OF (b)		USE OF (b)				
Frequency	Expected	Cell Chi-Sq.	Percent	Row Pct.	Column Pct.	
	1	2	3	4	5	Total
1	12	1	0	0	0	13
	1.76	2.44	2.30	3.79	2.71	
	59.56	0.85	2.30	3.79	2.71	
	12.50	1.04	0.00	0.00	0.00	13.54
	92.31	7.69	0.00	0.00	0.00	
	92.31	5.56	0.00	0.00	0.00	
2	0	7	0	0	0	7
	.95	1.31	1.24	2.04	1.46	
	.95	24.65	1.24	2.04	1.46	
	0.00	7.29	0.00	0.00	0.00	7.29
	0.00	100.0	0.00	0.00	0.00	
	0.00	38.89	0.00	0.00	0.00	
3	1	9	5	4	0	19
	2.57	3.56	3.36	5.54	3.96	
	0.96	8.30	0.79	0.43	3.96	
	1.04	9.38	5.21	4.17	0.00	19.79
	5.26	47.37	26.32	21.05	0.00	
	7.69	50.00	29.41	14.29	0.00	
4	0	1	11	10	2	24
	3.25	4.5	4.25	7	5	
	3.25	2.72	10.72	1.29	1.8	
	0.00	1.04	11.46	10.42	2.08	25.00
	0.00	4.17	45.83	41.67	8.33	
	0.00	5.56	64.71	35.71	10.00	
5	0	0	1	14	18	33
	4.47	6.19	5.84	9.63	6.88	
	4.47	6.19	4.04	1.99	18.00	
	0.00	0.00	1.04	14.58	18.75	34.38
	0.00	0.00	3.03	42.42	54.55	
	0.00	0.00	5.88	50.00	90.00	
Total	13	18	17	28	20	96
	13.54	18.75	17.71	29.17	20.83	100.0

Frequency Missing = 1

Chi-Square Statistic P=.0000

Table C-3

Chi-Square Test - Appeal of Application (c) by
Use of Application (c)

APPEAL OF (c)	USE OF (c)					
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	1	2	3	4	5	Total
1	14	0	1	0	0	15
	2.81	2.5	3.13	3.28	3.28	
	44.50	2.5	1.45	3.28	3.28	
	14.58	0.00	1.04	0.00	0.00	15.63
	93.33	0.00	6.67	0.00	0.00	
	77.78	0.00	5.00	0.00	0.00	
2	3	6	2	0	0	11
	2.06	1.83	2.29	2.41	2.41	
	0.43	9.47	0.04	2.41	2.41	
	3.13	6.25	2.08	0.00	0.00	11.46
	27.27	54.55	18.18	0.00	0.00	
	16.67	37.50	10.00	0.00	0.00	
3	1	7	7	2	0	17
	3.19	2.83	3.54	3.72	3.72	
	1.50	6.13	3.38	0.79	3.72	
	1.04	7.29	7.29	2.08	0.00	17.71
	5.88	41.18	41.18	11.76	0.00	
	5.56	43.75	35.00	9.52	0.00	
4	0	3	3	8	4	18
	3.38	3	3.75	3.94	3.94	
	3.38	0	0.15	4.19	0.00	
	0.00	3.13	3.13	8.33	4.17	18.75
	0.00	16.67	16.67	44.44	22.22	
	0.00	18.75	15.00	38.10	19.05	
5	0	0	7	11	17	35
	6.56	5.83	7.29	7.66	7.65	
	6.56	5.83	0.01	1.46	11.40	
	0.00	0.00	7.29	11.46	17.71	36.46
	0.00	0.00	20.00	31.43	48.57	
	0.00	0.00	35.00	52.38	80.95	
Total	18	16	20	21	21	96
	18.75	16.67	20.83	21.88	21.88	100.0

Frequency Missing = 1

Chi-Square Statistic P=.0000

Table C-4

Chi-Square Test - Pay for Initial Scan by Pay for Update Scan

INITIAL	UPDATE			
Frequency				
Expected				
Cell Chi-Sq.				
Percent				
Row Pct.				
Column Pct.	\$0	\$5	\$10	Total
\$0	33	1	0	34
	16.61	16.21	1.19	
	16.19	14.27	1.19	
	38.37	1.16	0.00	39.53
	97.06	2.94	0.00	
	78.57	2.44	0.00	
\$5	9	24	0	33
	16.12	15.73	1.15	
	3.14	4.34	1.15	
	10.47	27.91	0.00	38.37
	27.27	72.73	0.00	
	21.43	58.54	0.00	
\$10	0	16	3	19
	9.28	9.06	0.66	
	9.28	5.32	8.24	
	0.00	18.60	3.49	22.09
	0.00	84.21	15.79	
	0.00	39.02	100.0	
Total	42	41	3	86
	48.84	47.67	3.49	100.0

Frequency Missing = 11

Chi-Square Statistic P=.0000

Table C-5

Chi-Square Test - Fit Problems by Pay for Initial Scan

FIT PROBLEMS		INITIAL		
Frequency				
Expected				
Cell Chi-Sq.				
Percent				
Row Pct.				
Column Pct.				
	\$0	\$5	\$10	Total
No (0)	8	2	2	12
	4.58	4.72	2.70	
	2.55	4.57	0.18	
	8.99	2.25	2.25	13.48
	66.67	16.67	16.67	
	23.53	5.71	10.00	
Yes (1)	26	33	18	77
	29.42	30.28	17.30	
	0.40	0.24	0.03	
	29.21	37.08	20.22	86.52
	33.77	42.86	23.38	
	76.47	94.29	90.00	
Total	34	35	20	89
	38.20	39.33	22.47	100.0

Frequency Missing = 8

Chi-Square Statistic P=.0840

Table C-6

Chi-Square Test - Fit Problems by Pay for Update Scan

FIT PROBLEMS		UPDATE			
Frequency					
Expected					
Cell Chi-Sq.					
Percent					
Row Pct.					
Column Pct.		\$0	\$5	\$10	Total
No (0)	8	5	0	13	
	6.13	6.43	.45		
	0.57	0.32	.45		
	9.20	5.75	0.00	14.94	
	61.54	38.46	0.00		
	19.51	11.63	0.00		
Yes (1)	33	38	3	74	
	34.87	36.58	2.55		
	0.10	0.06	0.08		
	37.93	43.68	3.45	85.06	
	44.59	51.35	4.05		
	80.49	88.37	100.0		
Total	41	43	3	87	
	47.13	49.43	3.45	100.0	

Frequency Missing = 10

Chi-Square Statistic P=.4560

Table C-7

Chi-Square Test - Age by Most Appealing Application

AGE	APPLICATION			
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	(a)	(b)	(c)	Total
1. Below 30	5	6	5	16
	4	6.10	5.90	
	0.25	0.00	0.14	
	5.95	7.14	5.95	19.05
	31.25	37.50	31.25	
	23.81	18.75	16.13	
2. 30-39	9	8	13	30
	7.50	11.43	11.07	
	0.30	1.03	.34	
	10.71	9.52	15.48	35.71
	30.00	26.67	43.33	
	42.86	25.00	41.94	
3. 40-49	6	11	6	23
	5.75	8.76	8.49	
	0.01	0.57	0.72	
	7.14	13.10	7.14	27.38
	26.09	47.83	26.09	
	28.57	34.38	19.35	
4. 50+	1	7	7	15
	3.75	5.71	5.54	
	2.02	0.29	0.39	
	1.19	8.33	8.33	17.86
	6.67	46.67	46.67	
	4.76	21.88	22.58	
Total	21	32	31	84
	25.00	38.10	36.90	100.0

Frequency Missing = 13

Chi-Square Statistic P=.4170

(a) = Made-to-measure

(b) = Data card

(c) = Computer imaging

Table C-8

Chi-Square Test - Age by Most Usable Application

AGE	APPLICATION			
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	(a)	(b)	(c)	Total
1. Below 30	5	6	5	16
	3.39	6.78	5.84	
	0.77	0.09	0.12	
	5.88	7.06	5.88	18.82
	31.25	37.50	31.25	
	27.78	16.67	16.13	
2. 30-39	5	10	14	29
	6.14	12.28	10.58	
	0.21	0.42	1.08	
	5.88	11.76	16.47	34.12
	17.24	34.48	48.28	
	27.78	27.78	45.16	
3. 40-49	6	10	7	23
	4.87	9.74	8.39	
	0.26	0.01	0.23	
	7.06	11.76	8.24	27.06
	26.09	43.48	30.43	
	33.33	27.78	22.58	
4. 50+	2	10	5	17
	3.60	7.2	6.2	
	0.71	1.09	0.23	
	2.35	11.76	5.88	20.00
	11.76	58.82	29.41	
	11.11	27.78	16.13	
Total	18	36	31	85
	21.18	42.35	36.47	100.0

Frequency Missing = 12

Chi-Square Statistic P=.5120

(a) = Made-to-measure

(b) = Data card

(c) = Computer imaging

Table C-9

Chi-Square Test - Size by Most Appealing Application

SIZE	APPLICATION			
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	(a)	(b)	(c)	Total
1. Small	4	4	0	8
	2.13	2.94	2.94	
	1.65	0.39	2.94	
	5.06	5.06	0.00	10.13
	50.00	50.00	0.00	
	19.05	13.79	0.00	
2. Average	11	14	12	37
	9.84	13.58	13.58	
	0.14	0.13	0.18	
	13.92	17.72	15.19	46.84
	29.73	37.84	32.43	
	52.38	48.28	41.38	
3. Large	6	11	17	34
	9.04	12.48	12.48	
	1.02	0.18	1.64	
	7.59	13.92	21.52	43.04
	17.65	32.35	50.00	
	28.57	37.93	58.62	
Total	21	29	29	79
	26.58	36.71	36.71	100.0

Frequency Missing = 10

Chi-Square Statistic P=.0870

(a) = Made-to-measure

(b) = Data card

(c) = Computer imaging

Table C-10

Chi-Square Test - Size by Most Usable Application

SIZE	APPLICATION			
Frequency Expected Cell Chi-Sq. Percent Row Pct. Column Pct.	(a)	(b)	(c)	Total
1. Small	4	4	0	8
1.8		3.3	2.9	
2.69		0.15	2.9	
5.00		5.00	0.00	10.00
50.00		50.00	0.00	
22.22		12.12	0.00	
2. Average	10	17	10	37
8.33		15.26	13.41	
0.34		0.20	0.87	
12.50		21.25	12.50	46.25
27.03		45.95	27.03	
55.56		51.52	34.48	
3. Large	4	12	19	35
7.88		14.44	12.69	
1.91		0.41	3.14	
5.00		15.00	23.75	43.75
11.43		34.29	54.29	
22.22		36.36	65.52	
Total	18	33	29	80
	22.50	41.25	36.25	100.0

Frequency Missing = 9

Chi-Square Statistic P=.0130

(a) = Made-to-measure

(b) = Data card

(c) = Computer imaging

Table C-11

Chi-Square Test - Fit Problems by Amount Spent on Wardrobe

FIT PROBLEMS	AMOUNT SPENT				
Frequency					
Expected					
Cell Chi-Sq.					
Percent					
Row Pct.					
Column Pct.	a	b	c	d	Total
No (0)	2	7	5	0	14
	1.49	6.55	4.32	1.64	
	0.18	0.03	0.11	1.64	
	2.13	7.45	5.32	0.00	14.89
	14.29	50.00	35.71	0.00	
	20.00	15.91	17.24	0.00	
Yes (1)	8	37	24	11	80
	8.51	37.45	24.68	9.36	
	0.03	0.01	0.02	0.28	
	8.51	39.36	25.53	11.70	85.11
	10.00	46.25	30.00	13.75	
	80.00	84.09	82.76	100.0	
Total	10	44	29	11	94
	10.64	46.81	30.85	11.70	100.0

Frequency Missing = 3

Chi-Square Statistic P=.5140

a = Below \$200

b = \$200-499

c = \$500-999

d = \$1,000+

APPENDIX D
ANALYSIS OF VARIANCE AND MEANS

Table D-1

ANOVA Test - Appeal of Application (a) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	7.04	2.35	1.04	.3808
Error	93	210.74	2.27		
Corrected Total	96	217.77			

Table D-2

Appeal of Application (a) Means by Age Group

Group	N	Mean	Std. Err.
Below 30	18	3.39	1.46
30-39	35	3.71	1.54
40-49	25	3.64	1.47
50+	18	2.84	1.54

Table D-3

ANOVA Test - Appeal of Application (b) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	6.38	2.13	1.12	.3453
Error	92	174.77	1.90		
Corrected Total	95	181.16			

Table D-4

Appeal of Application (b) Means by Age Group

Group	N	Mean	Std. Err.
Below 30	18	3.39	1.38
30-39	35	3.71	1.43
40-49	25	3.64	1.29
50+	18	2.84	1.38

Table D-5

ANOVA Test - Appeal of Application (c) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	4.36	1.45	.66	.5764
Error	92	201.63	2.19		
Corrected Total	95	205.99			

Table D-6

Appeal of Application (c) Means by Age Group

Group	N	Mean	Std. Err.
<u>Below 30</u>	18	3.44	1.25
30-39	35	3.31	1.53
40-49	25	3.84	1.55
50+	18	3.38	1.50

Table D-7

ANOVA Test - Use of Application (a) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	6.94	2.31	1.45	.2341
Error	92	147.06	1.60		
Corrected Total	95	154.00			

Table D-8

Use of Application (a) Means by Age Group

Group	N	Mean	Std. Err.
Below 30	18	2.94	1.39
30-39	35	2.68	1.30
40-49	25	3.04	1.21
50+	18	2.27	1.12

Table D-9

ANOVA Test - Use of Application (b) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	5.81	1.94	1.07	.3652
Error	92	166.19	1.81		
Corrected Total	95	172.00			

Table D-10

Use of Application (b) Means by Age Group

Group	N	Mean	Std. Err.
Below 30	18	3.28	1.41
30-39	35	2.97	1.34
40-49	25	3.60	1.35
50+	18	3.28	1.27

Table D-11

ANOVA Test - Use of Application (c) by Age Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	3	1.03	.35	.17	.9186
Error	92	190.70	2.07		
Corrected Total	95	191.74			

Table D-12

Use of Application (c) Means by Age Group

Group	N	Mean	Std. Err.
Below 30	18	3.00	1.41
30-39	35	3.06	1.49
40-49	25	3.28	1.40
50+	18	3.11	1.41

100

Table D-13

ANOVA Test - Appeal of Application (a) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	.71	.36	.16	.8513
Error	86	189.47	2.20		
Corrected Total	88	190.18			

Table D-14

Appeal of Application (a) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	3.56	1.67
Average	42	3.55	1.56
Large	38	3.37	1.34

Table D-15

ANOVA Test - Appeal of Application (b) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	.32	.16	.09	.9152
Error	85	153.58	1.81		
Corrected Total	87	153.90			

Table D-16

Appeal of Application (b) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	3.67	1.32
Average	41	3.78	1.37
Large	38	3.66	1.32

Table D-17

ANOVA Test - Appeal of Application (c) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	1.81	.90	.43	.6502
Error	85	177.47	2.08		
Corrected Total	87	179.27			

Table D-18

Appeal of Application (c) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	3.66	1.32
Average	41	3.43	1.47
Large	38	3.74	1.45

Table D-19

ANOVA Test - Use of Application (a) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	1.00	.50	.31	.7355
Error	85	137.37	1.62		
Corrected Total	87	138.36			

Table D-20

Use of Application (a) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	3.11	1.45
Average	41	3.90	1.37
Large	38	2.76	1.01

Table D-21

ANOVA Test - Use of Application (b) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	1.47	.74	.42	.6584
Error	85	149.15	1.75		
Corrected Total	87	150.63			

Table D-22

Use of Application (b) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	3.00	1.58
Average	41	3.39	1.39
Large	38	3.45	1.18

Table D-23

ANOVA Test - Use of Application (c) by Body Size Group

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F
Model	2	4.51	2.26	1.13	.3292
Error	85	170.39	2.00		
Corrected Total	87	174.90			

Table D-24

Use of Application (c) Means by Body Size Group

Group	N	Mean	Std. Err.
Small	9	2.67	1.00
Average	41	3.14	1.48
Large	38	3.42	1.43

Appendix F

Student paper, Sen

**FINDING THE OPTIMUM NUMBER OF CASES REQUIRED
FOR
PREDICTING SHIRT SIZES
USING
REMIND: A CBR SHELL**

Submitted by: Raktim Sen

Submitted to: Dr. Steve Davis

Course #: MGT 918

Fall 1992

Acknowledgements

I acknowledge with thanks the help Dr Steve Davis, my instructor had extended towards me from time to time during the course of this study. I also thank Sarat, who helped me with his helpful hints.

This study has opened for me a new line of thinking-- thinking in terms of case-based reasoning. I thank Dr. Davis, my instructor, for giving me the opportunity to work on this project.

CONTENTS

Abstract.....	1
Introduction.....	2
Background.....	3
Case-Based Resoning.....	5
"REMIND".....	6
The Project.....	9
Limition of the Study.....	15
Conclusion.....	17

ABSTRACT

An expert system is being developed with "REMIND", a CBR shell, to predict the shirt sizes of soldiers. This study was done to find out the optimum number of cases required to predict satisfactorily. The study was done with 20 test cases. The study revealed that 1000 cases resulted in good prediction performance.

INTRODUCTION

An expert system is being developed at Clemson Apparel Research center under the guidance of Dr Steve Davis, for predicting the correct garment sizes for soldiers. This systems is being developed using "Case-Based Reasoning" (CBR) shell "REMIND". Case-based reasoning relies on the outcome of previously stored cases to predict an outcome for a case which is similar or nearly similar to one or more stored cases. It seems logical to infer that more the number of cases, better will be the accuracy of prediction.

In other words, it is expected that as cases are added, the better will be the output prediction, but at some point the marginal improvement may become negligible. This paper aims at determining the point where the "learning curve" starts flattening.

BACKGROUND

Soldiers need uniforms, and they need uniforms of the right size. Currently the U.S Army uses a manual method of assigning sizes. This method involves a fitter, who takes the body measurements of a soldier and assigns sizes for the different garments to be issued. If the assigned is not correct, the final size is determined by trial and error method.

Inaccuracies in predicting the correct garment sized could be due to many reasons. The fitter may take wrong measurements due to wrong placement of measuring tape, variations in tension of the measuring tapes, or due to fatigue. Secondly, even if the measurements were taken correctly, predicting the size means converting these body measurement into standard sizes, and the fitter could make a wrong decision. Finally, even if the body measurements are correct, predicting the correct size would involve accounting for the priorities among the measurements taken.

A prototype expert system was developed under Dr. Steve Davis and tried out at Fort Jackson. This system had to rely on body measurements taken manually. But its success, both in terms of saving time and money, has led to the current project of automating the entire process of size predicting.

The total system can be divided into two parts: (1) Taking the

correct body measurements, and (2) Predicting the right garment size. This student was involved with one of the aspects (testing) of the second parts of the project.

CASE BASE REASONING

Case-based reasoning is an emerging AI technology. It uses past experiences or cases to solve current problem¹. A rule-based expert system solves problems by taking input specifications usually through a question-and-answer dialogue. These input are then chained together to the appropriate set of rules from the rule-base to arrive at a solution (Figure 1).

Case-based reasoning operates in very different way. Given the input specifications, a case-based reasoning system will search for a case which is exactly similar to the input case. If successful it will give the solution directly. If not, it will retrieve a case that is nearly as similar. This solution may not be entirely appropriate. The human user then has to modify small portions of the retrieved case in what is known as "case adaptation". The result of case adaptation is the completed solution. The new solution may be stored as a case for future references (Figure 2).

¹ Baarletta, R. "An Introduction to Case Base Reasoning". AI Expert, August, 1991.



Figure 1. How a rule-based expert system works.

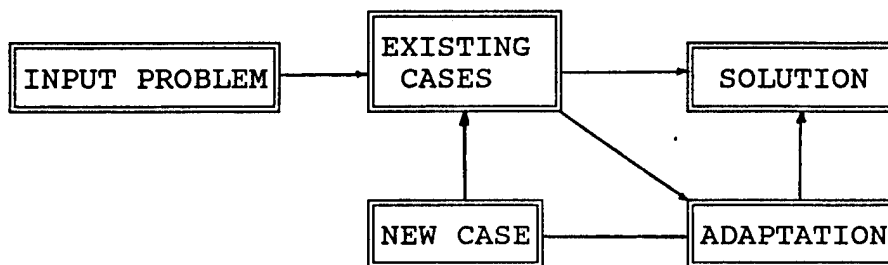


Figure 2: How a case-based system works.

"REMIND"

The CBR shell used for the development of the expert systems is "REMIND", a CBR development shell. In the following paragraphs the basics of the shell will be discussed vis-a-vis, the project.

Cases are represented in REMIND using four of the nine editors that are available. These four editors, viz., The field editor, the symbol editor, the formula editor, and the QModel editor, allow users to define a case and knowledge associated with it.

FIELD EDITOR

The field editor allows the user to define the fields that make up the description of the case. Fields can be of various types. The simple fields are text, integers, booleans, dates, and real numbers. More complex field types are symbols, lists, cases, and formulas. Symbol fields allow users to arrange information in a general to specific hierarchy (e.g. **matter**-->**solid**, **liquid**, **gas**; **liquid**-->**acidic**, **alkaline**, **neutral**; **acidic**--> **nitric acid**, **sulphuric acid**, etc.). Symbols may also be used to define subjective, partial ordering relationships (e.g. **big**--> **medium**--> **small**).

The list fields allow user to create lists of individual features (e.g. list of accessories in a car). The case fields allow the user to link together cases in the same library or another case library (e.g. noting down the performance of a new machine every month for six months and then linking together these cases to find out the average performance over the six months).

period).

SYMBOL EDITOR

Symbol fields are used to represent conditions, states, ratings, descriptions and other symbolic data, and the possible field values are stored as symbol hierarchy. The symbols are arranged in a parent-child relationship. The symbol editor allows editing of symbol hierarchy.

FORMULA EDITOR

The formula editor allows the user to create derived fields from existing fields in the case base. The user may use fields, symbol values and constants.

QMODEL EDITOR

The QModel (qualitative model) editor allows the user to graphically express known causal relationship.

Once the fields have been defined, the user uses a form defined in the form editor to input cases. Alternatively, cases may be imported from an existing database using data import editor.

Once the cases have been imported, the user must now decide on one of the two indexing types REMIND offers, viz., inductive indexing and nearest neighbor indexing. Inductive indexing is used where a definitive outcome field can be specified, e.g. shirt size

of a person. It is possible that the user may have the measurements of the entire body of a person but knows that only few of these contribute to deciding the correct size. These fields should be the "match" fields whereas the shirt size should be the "outcome" field.

After the "outcome" and "match" fields have been designated, a cluster is build using the cluster editor. The cluster tree is basically a decision tree for discriminating between cases of different outcomes.

In Nearest Neighbor Indexing, REMIND performs a similarity match between the features of two cases. Thus if one is looking for a house with 3 bedrooms, 2 bathroom, a lawn and a garage for two cars, then house matching all these aspects is a perfect match. But if there is no perfect match, the nearest neighbor would be the one which matches the requirement in most of the ways. The user is allowed to assign weights to the different fields so that the similarity is in the right perspective.

THE PROJECT

20 fields were used to represent a case. A case in our study is the body measurements of individual soldiers and the assigned garment sizes. The Field names and types are given in Figure 3. Four types of fields were used, viz., text, integer, real and symbol. The symbols were defined using the symbol editor. The five symbols used were S, M, R, X, and XL.

After the fields are created, cases can be entered. Since cases were imported, further preparation was needed. The steps are discussed below.

CASES TO BE USED FOR TESTING

In REMIND a case can have one of the three dispositions:

- 1) Stored case : - These are cases which will be retrieved. This means that these cases will be referred to and retrieved if acceptable as outcome.
- 2) Hypothetical Case: These are cases created by the user or a case whose outcome is required. REMIND will search the case base for a similar stored case. Hypothetical cases are not used for retrieval unless their status is changed to stored case.
- 3) Unstored Case : - These cases are primarily used to test the accuracy of the cluster tree. Since the outcome is already known, the quality of the prediction can be evaluated.

<u>Field Name</u>	<u>Filed Type</u>
First Name	Text
Last Name	Text
Head	Real
Neck	Real
Chest	Real
Sleeve	Real
Hips	Real
Waist	Real
Height	Integer
Weight	Integer
Cap	Real
Black Coat Length	Symbol
Black Coat Size	Integer
Green Coat Length	Symbol
Green Coat Size	Integer
Long Sleeve Shirt Size	Real
Long Sleeve Shirt Sleeve	Integer
Short Sleeve Shirt Size	Real
Trouser Length	Symbol
Trouser Size	Integer

Figure 3. Fields and Field types used

Our database had about 2500 cases. The first task was to determine which cases would be the unstored cases and be used for testing. It is possible in REMIND to set a percentage from the hypothetical cases to be randomly disposed as stored cases (the rest being assigned unstored status). But it was decided to have the same test cases tested with different number of stored cases. The assumption being that variations would be easier to explain with the same sample set. Thus it was decided to isolate the cases which would be used as test cases.

STRATEGY FOR TESTING

The 2500 odd cases were sorted alphabetically by last name. Therefore it may be assumed that the garment sizes were not appearing in any order. Instead of randomly choosing the test cases, every 100th case was chosen. From these 100 cases 20 cases were chosen to form the test cases. From the remaining, 2400 odd cases, the first 2000 cases were to form the case base, and from these cases, stored cases would be selected. Isolating the test cases would ensure that the same case is not retrieved as a solution.

The strategy was to start with 400 cases and then increase the size of the case-base by 200. These cases would be used to build a cluster tree. Then the 20 test cases would be imported and disposed as unstored cases. Next these unstored cases would be tested against the cluster tree using the REMIND Test Cluster function.

IMPORTATION OF DATA

The first step in importing data is to export the data to a flat text file. This was done in two steps. First, record # 1 to 400 (or the number of cases to be imported) were exported to a text file from dBase IV. This was done using the following command from the dBase dot prompt:

```
COPY TO filename.txt NEXT 200 TYPE DELIMITED.
```

This places a comma (,) as a delimiter, and all character fields would be between " ". The next step was to go to a DOS text editor to get rid of the " " from all the character fields. This was done using EMAX editor.

Once the data file is prepared, Data Import is selected from the Editor menu, and then from the Import menu, the raw data file is selected. At this point, the data from the first record can be seen in one of the windows of the Data Import Editor. The next step is to build an import map. This is done by placing the field tile and the raw data tile and linking them with appropriate translation formulas. The translation formulas were different for the four different types of fields used. For example, the text field "First Name" was linked with field # 2 of the raw data, the delimiter being a comma (,). The real number field "Head" was linked to raw field # 4 of the raw data using translation formula "Number From". The integer field "Height" was linked to the raw field # 3 using translation formula "Round" and "Number From". The symbol field "Black Coat Length" was linked using "First", "Find

Symbol named (raw field #) under root (length). In all cases comma (,) was used as the delimiter (as the text file had commas as delimiters).

Once the import map was made, "Import" was selected from the import sub-menu to start importation. (Actually the format was saved for using whenever importation had to be done.) After importation was complete, the "outcome" field and "match" fields were designated from the Importance editor. Since this student decided to test the prediction of Short sleeve shirt sizes, "Short Sleeve Shirt Size" was selected as the outcome field, and "Height", "Chest", "Neck", "Sleeve" and "Weight" were selected as "Match" fields. Other measurements were ignored as they were thought to play no part in determining shirt sizes.

The case editor was then selected and random disposition of cases were set at 100% so all cases became stored cases. Once stored, the "Build Cluster tree" was selected from the "Node" menu of the cluster editor. Minimum number of cases to split was selected to be 10 so that at least 10 cases were required to make a split. "Breadth First" was chosen so that the tree was build along the breadth first.

After the clustering was complete, a second set of data were imported, this time the 20 test cases. These cases were manually assigned "unstored" status and then the "Test Cluster" function was

used from the "Node" menu of the cluster editor. Minimum number of cases to retrieve was assigned to be 5. REMIND now tested the cluster and reported on the performance of retrieval under the following headings:

Similarity: Number of input cases (20 in our case), Percent of cases with similarity between 100 and 90%, 90 and 80 %, 80 and 70% and so on. (refer figure 4).

Next the value of the outcome field of the unstored cases (the 20 test cases) were changed to zero. The cases were again tested with the clusters. This time since no cases would match the outcome zero, all retrievals were unsatisfactory, but this testing has an advantage in that REMIND reports a case-wise summary of outcomes predicted. So if the outcome values of the test cases are known, they can be compared with the predicted outcome.

This method of double testing was done for stored case sizes of 400, 600, 800, 1000, 1200, 1400, 1600, 1800, and 2000. It must be noted here that, as the number of cases increase, the time to import and cluster increases significantly. The results are reported in Table 1 and 2 respectively.

The results show that when the number of stored cases is around 1000, the learning curve tends to become flat for prediction with similarity between 90 and 100%. Another interesting results was noticed. Test cases which were predicted correctly with a particular number of the stored cases, were correctly predicted

=====Check Function =====

Retrival under Inde Root
*** Overall Performance ***

	Input Cases		
Similarity:	Number	Percent	Total %
-----	-----	-----	-----
100 - 90%	10	50%	50%
90 -80%	3	15%	65%
80 - 70%	7	35%	100%

No cases had unsatisfactory retrival results.

Figure 4. REMIND Test Cluster Report.

Table 1

Overall Performance Clusters

Total # of Stored Cases: 400				Total # of Stored Cases: 1400			
Total # of Test (Unstored) Cases: 20				Total # of Test (Unstored) Cases: 20			
Similarity	Test Number	Input Cases Percent	Total %	Similarity	Number	Input Cases Percent	Total %
100 - 90 %	10	50%	50%	100 - 90 %	10	50%	50%
90 - 80 %	3	15%	65%	90 - 80 %	2	10%	60%
80 - 70 %	7	35%	100%	80 - 70 %	8	40%	100%
Total # of Stored Cases: 600				Total # of Stored Cases: 1600			
Total # of Test (Unstored) Cases: 20				Total # of Test (Unstored) Cases: 20			
Similarity	Number	Input Cases Percent	Total %	Similarity	Number	Input Cases Percent	Total %
100 - 90 %	9	45%	45%	100 - 90 %	11	55%	55%
90 - 80 %	1	5%	50%	90 - 80 %	3	15%	70%
80 - 70 %	9	45%	95%	80 - 70 %	6	30%	100%
70 - 60 %	1	5%	100%				
Total # of Stored Cases: 800				Total # of Stored Cases: 1800			
Total # of Test (Unstored) Cases: 20				Total # of Test (Unstored) Cases: 20			
Similarity	Number	Input Cases Percent	Total %	Similarity	Number	Input Cases Percent	Total %
100 - 90 %	9	45%	45%	100 - 90 %	12	60%	60%
90 - 80 %	2	10%	55%	90 - 80 %	3	15%	75%
80 - 70 %	8	40%	95%	80 - 70 %	4	20%	95%
70 - 60 %	1	5%	100%	70 - 60 %	1	5%	100%
Total # of Stored Cases: 1000				Total # of Stored Cases: 2000			
Total # of Test (Unstored) Cases: 20				Total # of Test (Unstored) Cases: 20			
Similarity	Number	Input Cases Percent	Total %	Similarity	Number	Input Cases Percent	Total %
100 - 90 %	11	55%	55%	100 - 90 %	11	55%	55%
90 - 80 %	3	15%	70%	90 - 80 %	3	15%	70%
80 - 70 %	6	30%	100%	80 - 70 %	6	30%	100%
Total # of Stored Cases: 1200							
Total # of Test (Unstored) Cases: 20							
Similarity	Number	Input Cases Percent	Total %				
100 - 90 %	11	55%	55%				
90 - 80 %	3	15%	70%				
80 - 70 %	6	30%	100%				

Case wise Analysls

Case #	True Size	Predicted size	# of Similar cases found for Stored Case Sizes of								
			400	600	800	1000	1200	1400	1600	1800	2000
1	16	16	10	1		3	9	4	8	6	3
		15.5	1	9	5	10		4	0		
		15									8
2	15.5	15.5	16	13	14	16	18	16	9	6	20
		16			3					1	
3	15.5	15	4	7	4	3	1	4	1	4	4
		15.5	3	1	5	4	6	14	8	9	11
		16							1	1	
4	15.5	15.5	30	19	28	8	13	9	9	4	12
		16					1			2	2
5	16	15.5		5	8	5	11	6	1	30	6
		16	7	1	1	1	14	15	3	44	2
		16.5							1	1	
6	15.5	15.5	30	9	8	10	8	6	9	4	9
		16			1	1				1	3
7	15.5	15	2					3			
		15.5	3	30	28	4	3	6	2	5	3
		16		11		5	3		3	2	
8	16	16	21	29	33	23	21	21	6	22	8
9	16.5	15.5		6	1		5	1	8	6	1
		16									5
		16.5	5		5	6		14	1	2	
		17				1		1			
10	16.5	16	7	4	11	37	4	5	1	7	14
		16.5		3	1	12	2	1	13	1	4
11	15.5	15	4	7	6	4		4	4	12	13
		15.5	3	1	4	5	10	4	1	14	18
12	16	15.5		2						2	
		16	13	5	11	18	5	16	2	3	7
		16.5					2		2		
		17						1			
13	15.5	14.5		3	4						
		15	24	3	4	16	11	16	17	19	18
		15.5									
14	14.5	14.5									
		15	24	24	19	23	14	6	12	2	15
		15.5					5	6	2	2	
		16						1	1		
15	15.5	15.5	18	26	12	34	25	11	11	21	25
16	17	16	1						2	1	
		16.5	12	9	4	4	12	21	7	16.5	3
		17	1	0	1	2				2	2
17	15.5	15.5	17	39	10	8	2	1	3	4	23
		16	1	3	1	1	5	6	2	5	
18	16	15.5	2					1			
		16	21	5	11	18	21	6	7	6	4
		16.5						1			
19	15.5	15.5					1				
		16	21	29	7	8	4	4	6	9	6
		16.5			2		5	1	3	1	
		17								1	1
20	16.5	16	5	5	9	11	8	11	12	1	16
		16.5								4	8

using almost any number of stored cases. This is evident from Table 2. We can see that case #s 2, 4, 6, 8, 12, 15, and 16 had no problem in finding a big number of similar cases. These cases could be termed as "good" cases, as they fit into a definite pattern. In sharp contrast, case #s 13, 14, 17, 19, and 20 can be termed as "bad" cases or out-liers. Case # 13 could not find a single match even from case base of 2000. In fact, similar cases were more consistently assigned size 15 (instead of 15.5). Same applies to the all the other bad cases. Then, there are cases which were consistently assigned different sizes with similar body measurements. (Case #s 1, 5, 7, & 17 assigned sizes 15.5 and 16, case #s 3 & 11 assigned sizes 15 and 15.5, case # 9 & 10 assigned sizes 16 and 16.5).

The above shows that there is a great deal of subjectivity in assigning garment sizes manually. The inconsistency is sometimes so much that it seems that sizes are assigned sometimes arbitrarily. Sometimes it may happen that the fitter finds it impossible to give due priority to a particular body measurement and assigns a "wrong size".

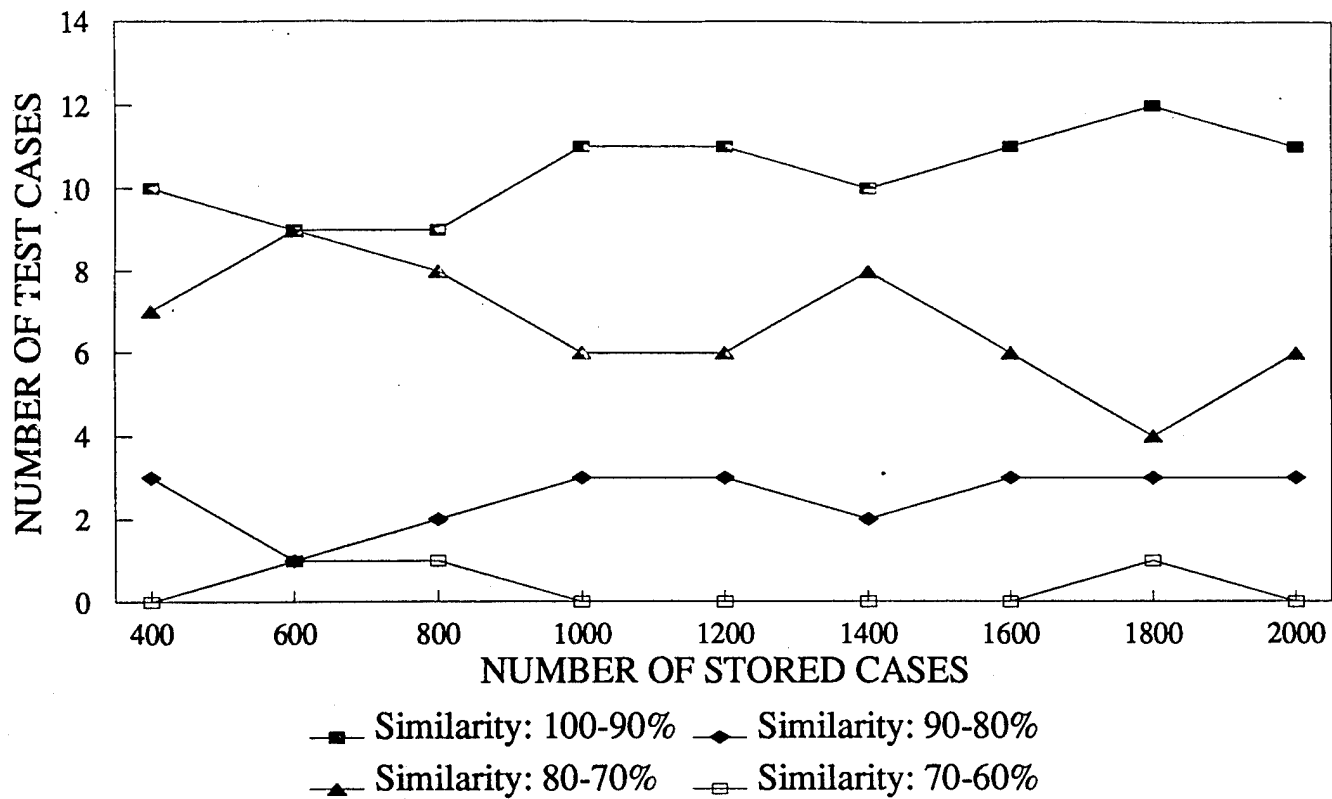
It is obvious that if the five bad cases are removed from the test cases, the level of predictions would be better.

THE LEARNING CURVE

From the observation, the learning curves were plotted Figure

TEST CLUSTER PERFORMANCE

FIGURE 5



5). It seems that the curve for 90 - 100% similarity tends to flatten out from 1000 cases. Thus the optimum number of cases required to predict short sleeve shirts may be around 1000.

LIMITATION OF THE STUDY

The main limitation in conducting the tests was the time the software takes to do almost any thing: right from opening a case library upto closing it. Also, disposition of unstored cases had to be done manually, as the system does not have the present capability to dispose all hypothetical cases as unstored cases. This had forced this student to keep the size of the unstored cases to a manageable size of 20. It is a possibility that if the size of the unstored cases can be increased to say 100, more meaningful results may be derived.

It may be noted that even with a small case base, the predictions are not too bad. (This student tried the tests with case base of 100 and 200). The reason could be that since the army has certain physical standards, the out liers (these out liers are different from the out liers discussed earlier in relation to assigning the correct sizes) do not make it. Thus, those make it fit certain norms and standard of physical measurement.

Finally, REMIND does not have any capabilities to print out reports. This makes it very difficult to read results and compare them. Very often, the reports overflow breadth-wise to the next screen. If at future date, printing capabilities are available, the clusters trees, with different size of cases, can be studied more efficiently to derive more meaning.

While, the whole concept of case-based reasoning, is very interesting and should become more popular, the present version of REMIND, though very convincing, is too slow.

CONCLUSION

The initial assumption, that there would be a learning curve which would indicate a optimum number of cases in the cases base proved to be correct. The tests showed that about 1000 cases in the case bases can give very satisfactory results. Since the test cases had 5 out of 20 (25 %) bad cases, it may be right to assume that a good percentage of the case base itself has bad cases. These cases will come to surface with repeated tests with different samples of larger size. Once these bad cases are identified, and removed, then the level of prediction would be a lot better even with a smaller case base.

Appendix G

Technical paper submitted, OLE

Automatic Measurement Extraction for Apparel from a 3D Body Scan

Roy P. Pargas, Nancy J. Staples, J. Steve Davis

Clemson Apparel Research

500 Lebanon Road, Pendleton, SC 29670

(Submitted to the Journal of Optics and Lasers in Engineering)

Abstract

This paper describes a project involving computer assisted measurement extraction from a 3D non-contact full-body scan. It explains the motivation behind the project, describes briefly the research plan, and provides details on the measurement extraction software currently being developed. The software provides the user with tools to take measurements from a digitized image. In addition, a measurement extraction language allows the user to develop macros, or short programs, designed to automate the measurement extraction process. The paper concludes with a description of the current status of the project.

1 Introduction

For as long as people have been wearing manufactured clothing, the apparel industry has relied on body measurements taken by a person with a measuring tape. Manual measurement is tedious, inconsistent and inaccurate. Different people may take measurements differently due to variation in 1) the compaction of flesh during measurement, 2) landmarking (determining body points which must be touched to be located), and 3) the tension of the measuring tape. Even measurements taken by the same person could lack consistency over the course of a day when that person gets tired. The availability of full-body scanning devices will make possible the capturing of X -, Y -, Z -data points representing the surface of the human body. With the appropriate software to convert these data points to body dimensions, highly accurate anthropometry (measurement of the body) will be possible.

A faster and more accurate method of body measurement could benefit both made-to-stock and made-to-measure operations in the apparel industry. For example, such a method could make it practical to gather and periodically update a large database of consumer body measurements. Such a database could be used to develop better fit models for improved standard sizing of garments. It could also provide a basis for a computer system which selects the best fit for a person from available garment sizes. An improved, automated

measurement method could facilitate the creation of garment patterns specifically for individuals through made-to-measure alteration of existing patterns or through custom pattern development.

There is a good market for the aforementioned computer applications. A recent study found that 78.3% of women surveyed were somewhat or very interested in using an automated garment size prediction system if it were available in a retail store, and 73.8% were interested in using an automated made-to-measure system (1). Fifty-seven percent of the women said they would be willing to pay to be measured with the aid of an automated 3D body scan system.

Military clothing initial issue points (CIIP), which typically provide clothing to several hundred soldiers a day, would also benefit from an automated size prediction system. A preliminary study by the authors at Fort Jackson, SC showed that even if such a system selected the correct size only 60% of the time, the reduction in time to fit 20,000 soldiers for trousers alone would be 100 hours per CIIP employee, or \$1000 each based on an average of \$10/hour. These savings could be realized at a single installation during the first year.

2 Objective

The objective of this study is to develop: 1) the software for extracting body measurements from the output of a 3D non-contact full-body scan, and 2) an expert system to predict men's U.S. Army dress uniform sizes for initial issue try-on. This paper focuses on measurement extraction.

3. Background

The technology for producing three-dimensional coordinates for the accurate computer representation of a scanned whole human body is currently being refined for practical use. The existence of this technology makes it possible to measure body dimensions more accurately than a human with a measuring tape. Since the body scanning technology is so new, no software exists for its application to the potential automation of current size selection and garment development methods. The usefulness of these technologies to the apparel industry for consumers is clear in the potential availability of better fit models for consumer products in stock sizes and the greater availability of made-to-measure or custom garments at an affordable price. The

manufacturers of consumer apparel products have been slow to respond to the potential of this new approach. This could be attributed to a resistance to change established practice, as well as the difficulty of supporting technological research and development in an era of declining profits and uncertain markets.

The U.S. military, however, is in the unique position of being both manufacturing contractor and consumer and can therefore readily appreciate the benefits of the application of advanced technology. The Defense Logistics Agency (DLA), the arm of the government which is responsible for the procurement of military uniforms, has shown its support by funding the research reported here.

The usefulness of these technologies to military purposes is evident. An efficient system of individual anthropometric data collection could provide a data base of anthropometry for many purposes including uniform development. This data could be teamed up with: 1) size prediction expert system software for faster, more accurate stock size uniform distribution, 2) software linkage to existing CAD pattern alteration software for made-to-measure uniform production, or 3) computer modeling of an individual body for custom pattern development.

In scenarios 2 and 3, single-ply numerically-controlled cutting and unit production could be added to automate the process further. Any of these applications could substantially reduce the average cost of providing dress uniforms to military personnel. Moreover, the body measurement data could be used to define better-fitting standard shapes for all military garments. Since only those body form variables affecting pattern shape are relevant to the determination of sizing categories or garment pattern development, irrelevant data could be automatically eliminated from consideration (the anthropologist's standard measurement set is not the same as that needed for garment development).

A data base of stored files could be accessed at any time for determining measurements specific to garment development. Information not readily accessible by manual means (mathematically-expressed posture and shoulder slope, maximum circumference of a combination of horizontal slices, body depth, asymmetry) could be gathered easily. If additional measurements are needed in the future, the files could still be available for reference (whereas humans cannot be quickly recalled for manual measuring). An additional benefit of this research is the exposure of the next generation of scientists to 3D technology for apparel uses.

4 Research Plan

The CIIP at Fort Jackson agreed to participate with Clemson Apparel Research (CAR) in the development and testing of any procedure that would expedite the clothing initial issue process. The Fort Jackson CIIP currently processes approximately 200 soldiers per day. The research was planned in two phases. In Phase 1 the tasks were: 1) to develop interactive software to make the output of a 3-dimensional, non-contact body scan useful for extracting body measurements, 2) to develop and calibrate expert system software which uses body dimensions as input and predicts the appropriate garment size for try on, and 3) with the cooperation of the CIIP staff, to test the expert system under conditions in which measurements have been taken by trained fitters. When the expert system was running smoothly, and the measurement extraction software developed, Phase 2 would include the incorporation of a 3-dimensional, non-contact measuring device to replace manual measuring. Researchers would work interactively through a computer interface with the measuring equipment to select the locations where body dimensions should be measured. These data would be electronically fed to the expert system and the size predictions would be printed out for each soldier. The time required for these operations and the level of prediction accuracy would be recorded and analyzed. At this writing, Phase 1 has been completed. However, since a field-test-worthy full-body scanner is not yet available, Phase 2 is ongoing.

In the United States the developers nearest to providing such a scanner are: Textile Clothing Technology Corporation, Cary, NC (white light system originally developed through Dimensional Measurement Systems, New York, NY by researchers at the New York Institute of Technology), Cyberware, Monterey, CA (laser-based systems currently being used primarily for head scanning), and Laser Design, Minneapolis, MN (laser-based systems currently being used for scanning complex engineered parts).

In the United Kingdom the National Engineering Laboratory (NEL) in Glasgow is developing a moire-fringe topography-based system for the Defence Clothing and Textiles Authority. Also involved in the NEL project is the Stores and Clothing Research and Development Establishment (the UK equivalent to the Defense Logistics Agency in the U.S.). A second research effort is being conducted by the University of Surrey. The scanning system used in the Surrey project is provided to the University of Surrey by the University

College Hospital (London) and is a laser-based system used to scan human heads. A third research effort is developing a scanning system, also laser-based, at the University of Loughborough.

Issues in the usefulness of full-body scanning include: 1) the ease of maintaining the posture of the subject, 2) the time exposure for the subject, 3) the time to create the model data set, and 4) the effort required to acquire further data. Table 1 compares these issues for classical, manual anthropometry (which is considerably more scientific and precise than the measurement of soldiers at a CIIP), laser scanning, and moire-fringe scanning. Although the moire-fringe characteristics reported in Table 1 seem to imply that its positive features would lead to implementation, the difficulties inherent in the synchronization of its views from the multiple cameras necessary have, thus far, prevented its usefulness. If aids can be devised to maintain the subject's posture for 15 to 20 seconds, then laser technology appears to be the closest to practical use.

Anthropologists, when preparing to measure the human body in their precise, scientific way, traditionally begin by "landmarking" the body, marking the location of anatomical guideposts which are evident only by touching the body to feel for 1) a bony protrusion under the soft flesh or 2) a joint where two bones are hinged together. Since it will not be possible to feel for these landmarks on a computer image of the surface of a human body, alternatives must be addressed. An acceptable means of locating some landmarks may be by defining mathematically an approximate location which, when the formula is applied consistently, will give sufficiently accurate data for garment development and size prediction purposes. For example, a substitute for the shoulder point, normally represented in manual mode by acromion (the bony protrusion at the end of the collar bone where it meets the upper arm) may be the leftmost point defined by the intersection with the contour of the body of a line which bisects the angle formed by the slope of [the leftmost points of the body slices defining] the upper arm and the slope of [the leftmost points of the body slices defining] the shoulder. Non-traditional aids may also be employed. A waistline for the wearing of clothing may be defined by placing a one-inch-wide elastic "belt" around the preferred location on the body before the scan. The resulting flesh compaction would then indicate the location to be measured. A key in the successful use of measurements obtained from body scans will be in the clear definition of where and how the measurements were taken. As long as any measurement data is accompanied by such documentation, users of data sets from multiple sources will be able to identify similarities and differences and not erroneously compare "apples and

oranges."

5 Measurement Software

Overview

One of the primary goals of this project is the development of software for extracting body measurements from the output of a 3D non-contact full-body scanner. There are several reasons for this goal. An automated system is more efficient than a human and achieves significant cost saving when a large number of bodies have to be measured, in a U.S. Army recruiting center, for example. The measurements generated by an automated system will be more consistent. A manual system depends entirely on the measuring skill of possibly many different people and the set of measurement data is almost certainly inconsistent. An automated measurement system may have commercial applications in apparel manufacturing and retail. One of the missions of Clemson Apparel Research is to provide support to the U.S. apparel industry through research and this project has clear potential benefits to this industry.

This section describes the measurement extraction software developed in the DLA study. Called 3DM, the software is written in C and runs under UNIX on a SUN workstation. It interprets a measurement extraction language (2) that is slowly evolving as the specific needs of the user become clear. The code utilizes X-window graphics libraries to manage the windowing environment. The code is still being developed. In particular, the measurement language described below continues to be improved.

The remainder of this section gives an overview of the input data format, the user interface, i.e., an overall view of how a user views and uses 3DM, a detailed description of each of the measurement and viewing functions, and an outline of the measurement language system currently being developed.

Input Data Format

The input data consists of points on the surface of the body, each point represented by three floating-point values corresponding to the point's X-, Y-, and Z-coordinates. There were ten sets of data used in this study. Each set consists of approximately eight thousand points. The portion of the body represented in each data set starts just above the collarbone and ends at the top of the thigh, including both arms. Two

of the datasets were female mannequins, one armless, the other with a single arm. The rest of the datasets were male forms.

User Interface

When initiated, 3DM activates five windows: a) Body Display, b) Control Menu, c) Image Name, d) Measurements, and e) Views. These are shown in Figure 1. The user initiates the measurement process by loading a figure which is exhibited in the *Body Display window*. Initially, a frontal view of the figure is shown. However the *Menu* allows the user to Rotate, Translate, and Scale the image. The user may select whether to rotate the image along *X*-, *Y*-, or *Z*-axes after selecting the desired angle of rotation. Translation along the *X*-, *Y*-, or *Z*-axes may be performed after the user sets the translation distance. Finally, the user may scale the image up or down by a preset percentage. Each of these instructions may be repeated, thus allowing the user total control in the placement and orientation of the figure.

Measurement

To take measurements of the image, the user clicks on the *Measure* button of the Menu. This opens the Measurements Menu, a list of four ways to measure the figure: *Circumference*, *Multipoint*, *Radial Measurement*, and *Language*.

The *Circumference* option allows the user to select a point on the body and to receive a circumferential measurement of the horizontal slice of the body containing the point. This measurement is useful in quickly obtaining such measurements as chest, waist, or seat. The measurement, expressed in inches, is displayed in the Measurements window. Figure 2 shows a chest measurement being taken. Any point on this horizontal slice may have been selected to obtain this measurement.

Multipoint allows the user to select a sequence of two or more points $P_1, P_2 \dots P_n$ on the surface of the image, and takes the sum of the surface distances between P_i and P_{i+1} for $i = 1, 2, \dots, n - 1$. This is equivalent to taking a tape measurement of the path from point P_1 to P_2 to P_3 and so on. The numerical sum of the measurements, expressed in inches, is displayed in the Measurements window. An example is shown in Figure 3 where the points selected were center back, left shoulder, left elbow, and left wrist. The

purpose is to take a measure of the length of the person's left arm. The result is a path on the surface of the body through the selected points. Following a smoothing of the path, the measurement is displayed in the Measurements window.

Radial Measurement allows the user to select a sequence of two or more points $P_1, P_2 \dots P_n$ on the surface of the image, and lists the $n - 1$ surface distances between P_1 and P_i for $i = 2, 3, \dots, n$. This is equivalent to taking a tape measurement from point P_1 to P_2 , from P_1 to P_3 and so on. The list of measurements, expressed in inches, is displayed in the Measurements window. Figure 4 shows a radial measurement taken from the center front point to each of the left shoulder, left arm pit, right arm pit, and right shoulder. The measurements are displayed in the Measurements window.

Slices and Views

The *Slices* option provides the user with three planar views of the figure. The user selects one or more points on the body and displays the selected planar slice of the figure in the *Views* window. The first view allows the user to fix the X -value. This view gives a profile of the body showing, for example, the curvature of the spine. The user may click on any point on the body, and the selected slice will be shown in profile in the *Views* window. In Figure 5, the user has selected a point near the center of the body. An outline of the slice is shown in the *Body Display* window and a profile view of the slice is shown in the *Views* window.

The second view allows the user to fix the Z -value, providing the user with a lateral slice of the figure. This may show an isolated view, for example, of the slope of the shoulders.

The third view allows the user to fix one or more Y -values and superimposes the horizontal slices one on top of another. This allows the user a top view of, for example, a person's waist and seat, superimposed on each other, allowing the user to trace the outer circumference of the resulting figure. In Figure 6, the user has selected two slices, the waist and the seat. Outlines of the slices are shown on the image in the *Body Display* window while a top view of the superimposed slices are shown in the *Views* window. This option is useful for defining and measuring the outer perimeter of multiple superimposed slices.

Measurement Language

Multipoint, Radial, Circumference Measurement and *Slices* each require the user to point and click at parts of the body and select the desired measurement. This is acceptable if the user has to measure only a small number of images. This process, however, becomes impractical if the user has to measure even a few dozen images. To overcome this problem, a *Language* is being designed to allow the user to write macros, i.e., short programs, to take any of the above measurements. This function allows the user to automate the process of measurement taking by defining, for example, how to recognize and measure a person's chest. The computer may then be instructed to take chest measurements of any number of images without human intervention. *Language*, therefore, provides the user with the power of *automatic measurement* of any number of parts of any number of body images. *Slice, Point, and Measurement*, each with several options.

A *Region* instruction is used to select a region of the body, where a region is one of the following: torso, left arm, right arm, left leg, right leg, upper body, and the entire body. A region must be selected before the user may issue *Slice* or *Point* instructions.

Slice and *Point* instructions continue the selection process. A *Slice* instruction isolates a specific slice within the region. A *Point* instruction allows the user to select a point within a slice. At any moment, focus is on a particular slice and point. However, slices and points may be saved and the user may issue instructions to continue the search for other slices and points. Saved slices and points may be used with *Measurement instructions*. Measurements may be saved for future computation or display.

Some slice instructions search for slices with special properties. For example, the *minslice* and *maxslice* instructions search regions for slices with the smallest or largest circumference. Related instructions are *topslice*, *middleslice*, and *bottomslice* which search for the top, middle, and bottom slices of a region. Focus may be controlled with *slicemove* instructions which instruct the computer to move from the slice currently being considered to another slice above or below.

Point instructions allow a user to select individual points within a slice. Two instructions available are *center* and *most*. *Center* takes as parameter one of the following: *FRONT, BACK, LEFT, RIGHT* which selects the center front, center back, center left, and center right points, respectively, of the current slice. *Most* takes the same four parameters and selects the foremost, hindmost, leftmost, and rightmost points of

the current slice. *MinX*, *MaxX*, *MinZ*, and *MaxZ* instructions select the points with the smallest *X*, the largest *X*, the smallest *Z*, and the largest *Z* values, respectively, between two specified points in the slice. Recall that all points in a slice have the same *Y* value. As with slices, the focus may shift from one point to another through *pointmove* instructions.

Three measurement instructions are available. *Circumference* takes a slice and returns the length of its circumference. *Directdistance* takes a sequence of points and returns the sum of the Euclidean distances between each two successive points. *Surfacedistance* takes a sequence of points and takes the sum of the surface distances between each two successive points. *Surfacedistance* is equivalent to the *Multipoint* option described above.

As mentioned earlier, these instructions may be used to write macros, i.e., short programs instructing the computer to take measurements automatically. For example, to take the chest measurement shown in Figure 2, one might use the macro shown in Figure 7.

The instruction *region torso* selects those slices of the body defined to be part of the torso, i.e., below the armpit and above the legs. The instruction *slicemove down 1.0* selects the slice closest to an inch below the topmost slice of the torso, approximately where the chest of a person is. The *slicesave* instruction saves the slice for the last instruction, *circumference*, which measures the length of the circumference of the chest slice.

Similarly, consider the macro in Figure 8 designed to measure sleeve length (see Figure 3). *Region upperbody* (line 2) selects those slices from the top of the image to the slice just above the legs. The computer is instructed to start with the top slice (line 3), move down about half an inch (line 4) and then to take and save the center back point (lines 5-6). The computer is instructed again to move down about one inch (line 7) before selecting and saving the left shoulder point. Next, we select the left arm region (line 10). The computer is instructed to move down about one-half of the distance of the arm and to select and save the point on the elbow (lines 11-13). The last point to save is that of the wrist, defined to be the left most point of the slice with minimum circumference in the arm (lines 14-16). Finally, line 17 takes the sleeve length.

It may be open to debate whether the two macros, as presented, provide a good way to measure chest circumference or sleeve length. There is also the question of whether such macros will extract the correct

measurement from image to image. This research is ongoing and experiments continue with different macros and figures. As new needs are uncovered, new instructions will be designed and included in the language.

6 Concluding Remarks

The development of 3DM continues. At present, the primary effort is in the expansion and improvement of the measurement extraction language. Experiments with different macros are being conducted to test the consistency and accuracy of the measurements taken. Although much has already been achieved in 3DM, this study has been hampered somewhat by the lack of a reliable full-body scanner. The images used in this study were generated by a prototype version of a moire-fringe scanner developed by Dimensional Measurement Systems, Inc. of New York City. DMS has since gone out of business. Since July 1993, development of its scanner has been undertaken by Textile and Clothing Technology Corporation, [TC]², of Cary, North Carolina. [TC]² expects to have a working scanner shortly. Cyberware, Inc. of Monterey, California and Laser Design of Minneapolis, Minnesota have also announced current efforts to develop full-body scanners, both laser-based. Each expects to have a scanner available in early 1995. When such a scanner eventually becomes available to the authors, much more rapid progress on the development of 3DM will occur. At that point, the authors will present an updated report on 3DM.

Acknowledgments

The body scan data used in this research was created by Dimensional Measurement Systems, Inc., New York, and provided by the Textile Clothing Technology Corporation, Cary, NC, current owner and developer of the former DMS equipment. This project has been supported by the U.S. Department of Defense, specifically the Defense Logistics Agency, under contract number DLA900-87-D-0017, DO 0026.

References

1. Knight, A. & Cassill, N., 3D Body Scanning Gets High Marks. *Apparel Industry Magazine*, (August 1994), 98-103.
2. Mak, R., Writing Compilers and Interpreters: An Applied Approach. John Wiley and Sons, Inc., New York, 1991.

August 2, 1994

	Classical Anthropometry	Laser Scan	Moire-fringe
Maintenance of subject posture	Easy	Difficult	Easy
Time exposure for subject	About 10 minutes but easy to "adjust"	About 17 seconds	About 1-2 seconds
Time to obtain data set	Quick to obtain limited data sets	Time to create models and extract measurements dependent on computational power of computer used	Time to create models and extract measurements dependent on computational power of computer used
Effort to acquire further data	Very difficult, subjects need recalling	Easy, computer model can be used many times	Easy, computer model can be used many times

Table 1. Comparison of classical anthropometry to scan technologies.

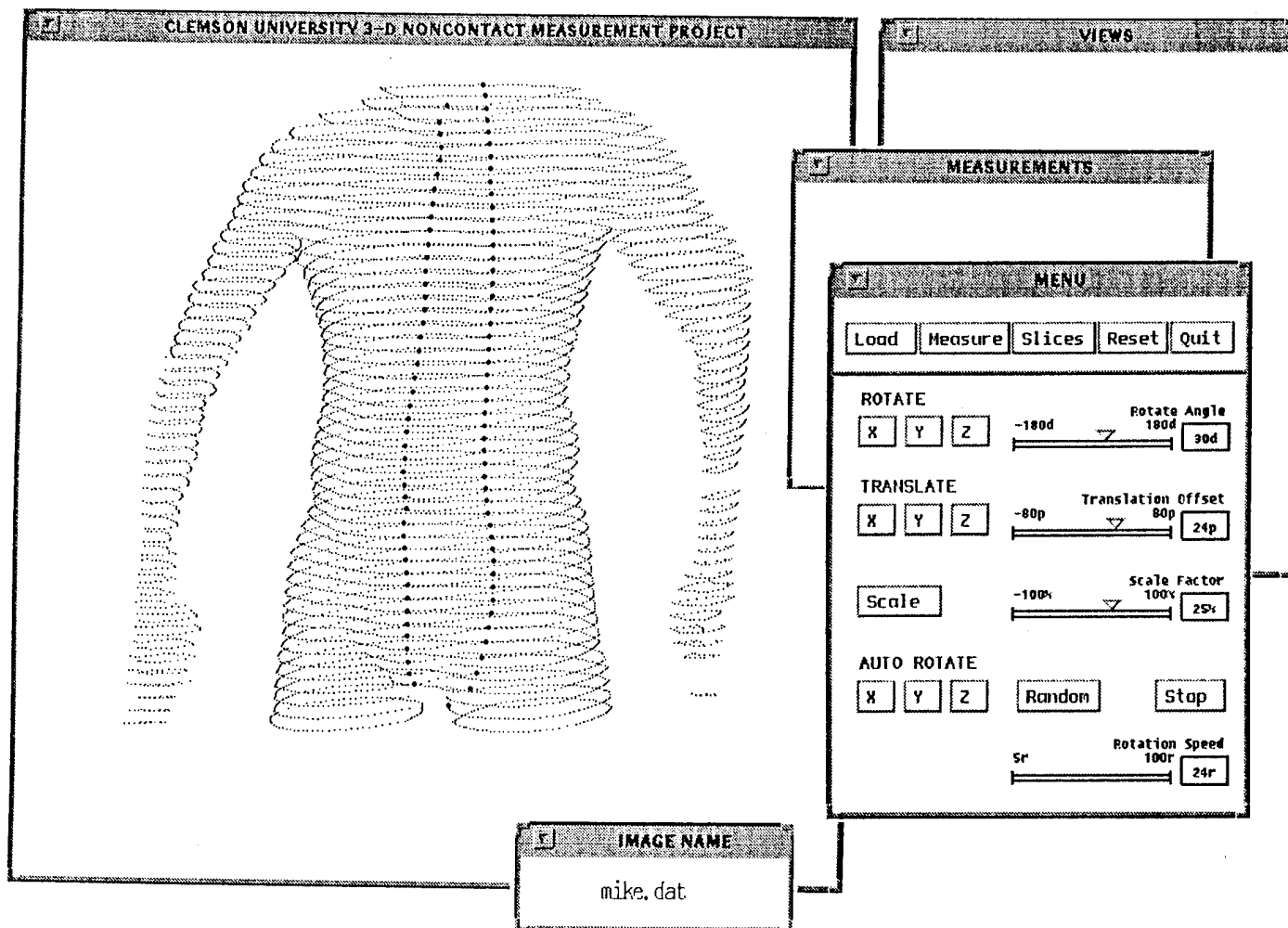


Figure 1. Five main windows used in 3DM.

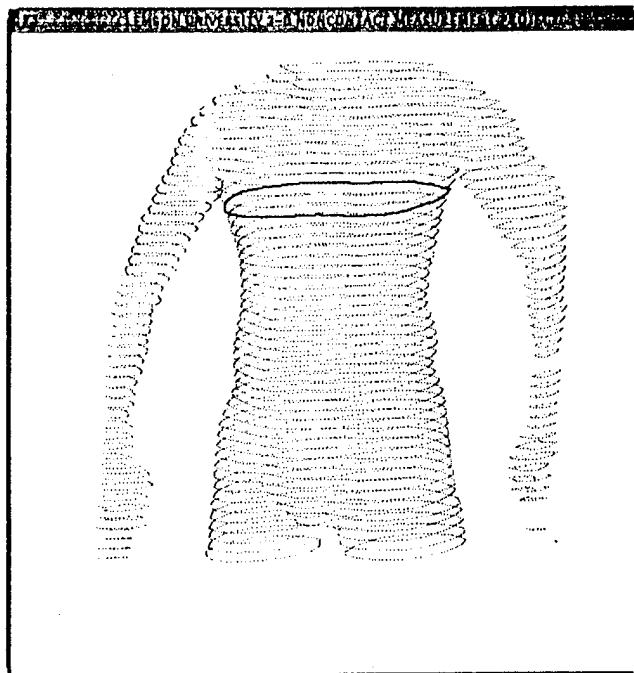


Figure 2. Measurement of chest circumference.

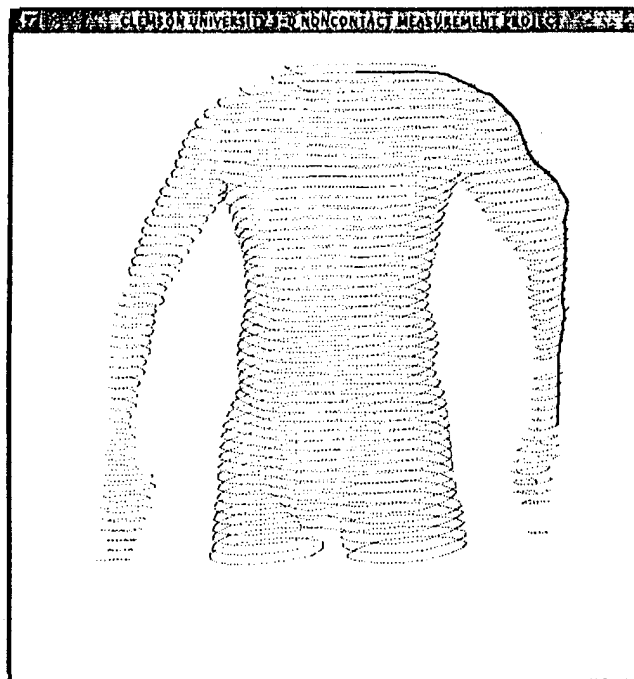


Figure 3. Measurement of arm length.

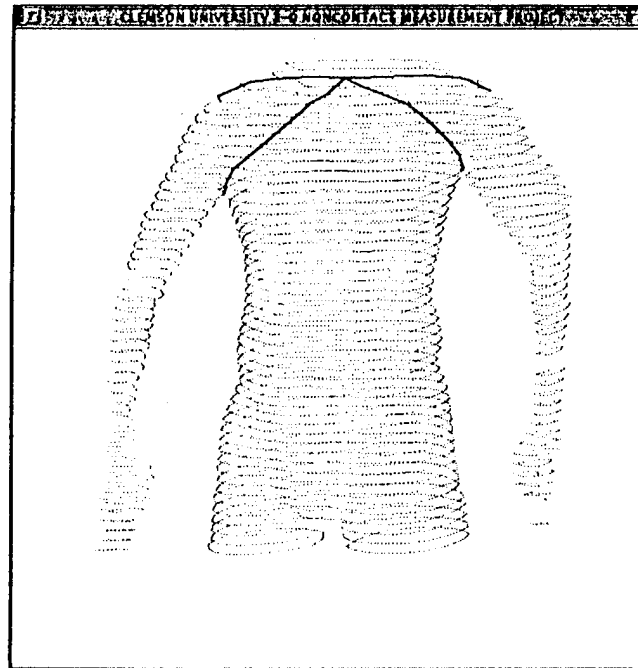


Figure 4. Radial measurement.

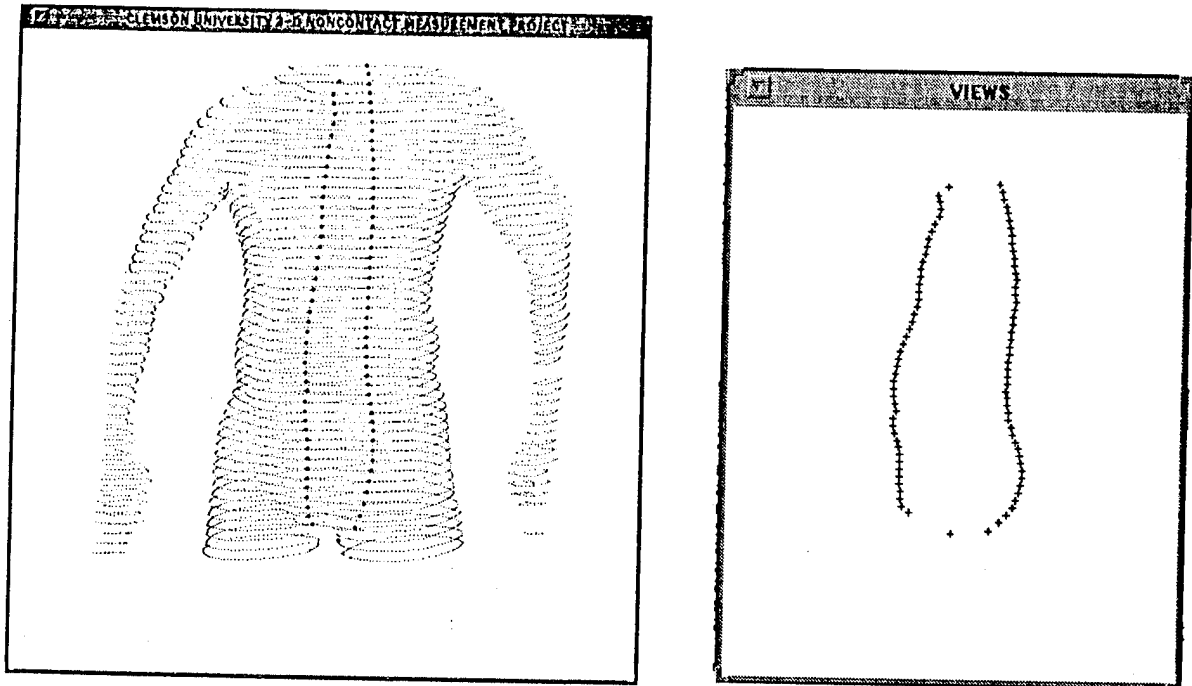


Figure 5. Planar view of a single slice of the figure.

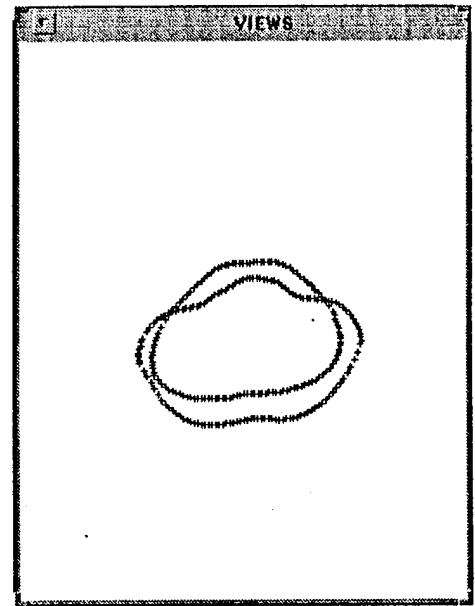
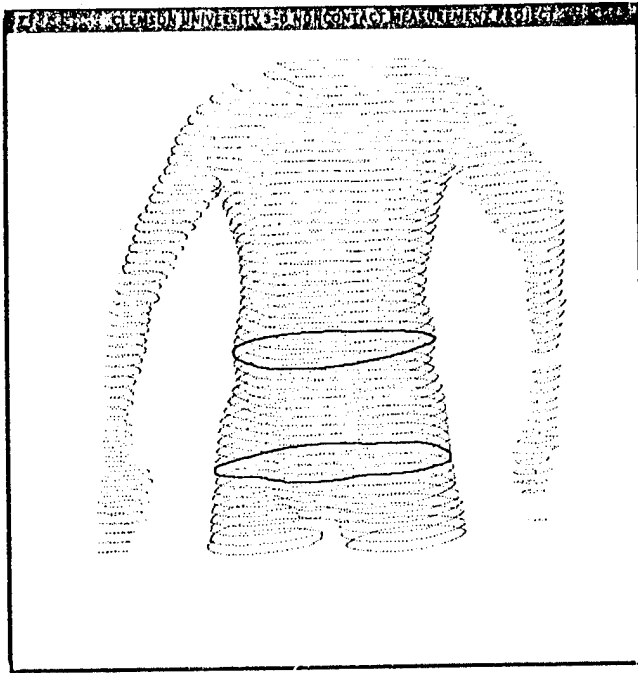


Figure 6. Top view of two horizontal slices superimposed on one another.

1. chest_measurement:
2. region torso
3. slicemove down 1.0
4. slicesave chest
5. circumference chest

Figure 7. Macro to measure chest circumference.

1. sleeve_length:
2. region upperbody
3. topslice
4. slicemove down 0.5
5. center back
6. pointsave center_back
7. slicemove down 1.0
8. most left
9. pointsave left_shoulder
10. region larm
11. selectslice 0.5
12. most left
13. pointsave left_elbow
14. minslice topslice bottomslic
15. most left
16. pointsave left_wrist
17. surfacedistance center_back left_shoulder
left_elbow left_wrist

Figure 8. Macro to measure sleeve length.

Appendix H

Technical paper submitted, IEEE

Predicting Garment Sizes with Case-Based Reasoning

J. Steve Davis, Roy Pargas and Nancy Staples, Clemson University

Selection of a Development Approach

The current manual system for selecting military uniform sizes employs expert fitters. In many instances the fitters choose the wrong size, which results in repeated, time-consuming trials of various sizes of garments. Because this system is labor-intensive for both fitters and soldiers, an automated approach might streamline the fitting process.

Two aspects of the manual system could be automated, the measurement and the size selection. This paper describes a project which investigated automated measurement but concentrated on developing a system for the Army to select sizes based on manual measurements. Automated measurement will be incorporated later when the necessary technology matures.

There appeared to be three possible foundations for a knowledge-based system which could automate the selection of garment sizes from manual measurements: 1) Army tables (which prescribe sizes for ranges of body measurements), 2) procedures used by human experts, or 3) case-based reasoning. We considered four ways to implement the system, which were applicable to certain of those foundations. We could develop a rule-based program or a case-based program, and with either approach employ a shell or not. These alternatives are outlined in Table 1, together with estimates of development speed and system accuracy, which are discussed next.

Table 1. Relative Development Speed and System Accuracy
of Possible Approaches to Development

	<u>Foundation for a knowledge-based system</u>		
	Army tables	Human knowledge	Case-based
<u>Implementation</u>			
Rule-based program			
with shell	quick, poor	medium, good	n/a
without shell	medium, poor	long, good	n/a
Case-based program			
with shell	n/a	n/a	quick, good
without shell	n/a	n/a	long, good

Since our project team had no experience implementing case-based programs, we initially gave serious consideration only to the other two alternatives. We decided to proceed initially toward a rule-based system (RBS) using a shell, because 1) that was the most commonly used approach for knowledge-based systems and our problem did not seem unusual, 2) experts were available, 3) our project team had experience with RBS, 4) a RBS shell was on hand, and 5) we could develop a prototype system faster with a RBS shell than with conventional programming. We figured that by developing a rule-based prototype, at least we would learn more about the problem domain. If the rule-based approach did not look good at that point we could adopt a different approach and take advantage of what we had learned.

The authors guided two graduate students who undertook development of a RBS as a masters degree project. To expedite system development, they decided to rely on the domain knowledge available in Army tables rather than to consult with expert fitters. The tables listed the clothing sizes appropriate for certain ranges of body measurements.

They chose VP-Expert, a simple shell, because they had learned how to use it in a course on expert systems and it seemed adequate for producing a prototype. To reduce the scope of the project they focused on trouser size prediction. Based on the Army tables, they developed a set of rules such as the following:

```
IF      Waist  >  25   AND
      Waist  <= 27   AND
      Seat   >  35.5 AND
      Seat   <= 36.5
THEN Size_26 = Yes;

IF      Size_26 = Yes  AND
      Inseam >  30   AND
      Inseam <= 32
THEN Size      = 26_Short
```

An informal evaluation of their system discouraged us about continuing to work toward a RBS solution. During their work we had learned from military fitters that the tables upon which the students' system was based were not a sufficient basis for size prediction. Expert fitters told us they seldom used them. We were concerned that to develop a properly functional RBS would require extensive research to determine how to convert expert knowledge into computer code. From interviews of expert fitters at the Fort Jackson, SC Army base and other military bases, we learned that knowledge acquisition for a rule-based system would be extremely difficult because 1) fitters employed a "you look like this size" approach and had difficulty describing precisely how they made decisions, 2) and different experts seemed to have different strategies.

Employing conventional programming didn't look very attractive either, for the same reasons. If we used the Army tables as a basis the resulting system would probably not predict sizes very well. To use human expertise as a basis would require difficult knowledge acquisition work. Either way, the heart of the coding

would probably be IF statements similar to the IF-THEN rules in the RBS prototype. Such a system would likely run faster than a RBS built with a shell, and we would have more control over the user interface. However, this kind of system would probably be more difficult to build and to change than one built with a shell. The ability to easily change the system was important. Military clothing styles are much more stable than are styles in the fashion industry and may be constant for several years, but occasionally they will be modified. For example, during the period of our project the Army changed the specifications for the long sleeve shirt, adopting multi-length sleeves, e.g. 32-33, and slight changes were made in the dimensions of other garments.

The students had produced an interesting system, but it helped us determine that neither a RBS nor a conventional program were attractive alternatives. Since styles were relatively stable, in between infrequent style changes, it occurred to us to explore case-based approaches, which are based more directly on the available data than are other methods. The main attraction was case-based reasoning (CBR) proponents' claim that knowledge engineering is simplified [1].

Selection of a Case Based Reasoning Shell

A CBR system includes a database of cases and a retriever. Cases are descriptions of previous problems and their outcomes (solutions). Given a new problem, the retriever finds the cases in the database which most closely resemble it. This project required a "problem solving" type of CBR system, which adapts old solutions to new problems, whereas an "interpretive reasoner" CBR system uses cases to help classify or pass judgement on new situations, as lawyers use previous legal cases to argue that a new situation belongs in a certain category [2]. A problem solving system can merely propose the solution associated with the most appropriate retrieved cases (which worked fine for this system), or can employ a subroutine to derive a recommendation based on the current problem and the retrieved cases.

We considered developing a case-based system from scratch, and figured that would probably involve using a database management system for storing and retrieving cases. To make retrieval speed fast enough with a large database, we might have to set up special indices to speed up retrieval. Also, we would have to code routines for calculating similarity of cases. Since the aforementioned steps could take considerable project time we were attracted to using one of the newly available shells for case-based system development, which had built-in tools for calculating similarity of cases and setting up indices. The project selected the ReMind CBR shell by Cognitive Systems, because it is easy to use, provides automatic indexing of cases, and facilitates rapid prototyping.

Development of a Database of Cases

A case-based system depends upon an adequate database of cases. We arranged for the clothing issue facility at Fort Jackson, South Carolina to provide records of the fitting of 4100 soldiers during 1992. Each record constituted a case, and consisted of a set of body measurements together with the garment sizes for an individual. The measurements were height, weight, head, neck, chest, waist, hips, and sleeve length (Figure 1). The garments were short sleeve shirt, long sleeve shirt, trousers, dress coat, and overcoat. The written records were entered into dBase IV files and checked for validity. The database of cases was converted from dBase IV into ASCII format and then imported into ReMind.

Scheme for Retrieving Similar Cases

A decision had to be made on how to handle the composite sizes of the coat, trousers and long sleeve shirt. Each consists of a numeric size and a length. The sleeve length is numeric, and the other lengths are symbolic (XS,S,R,L,XL). A composite size could be entered into a single field of a case, or the numeric size and

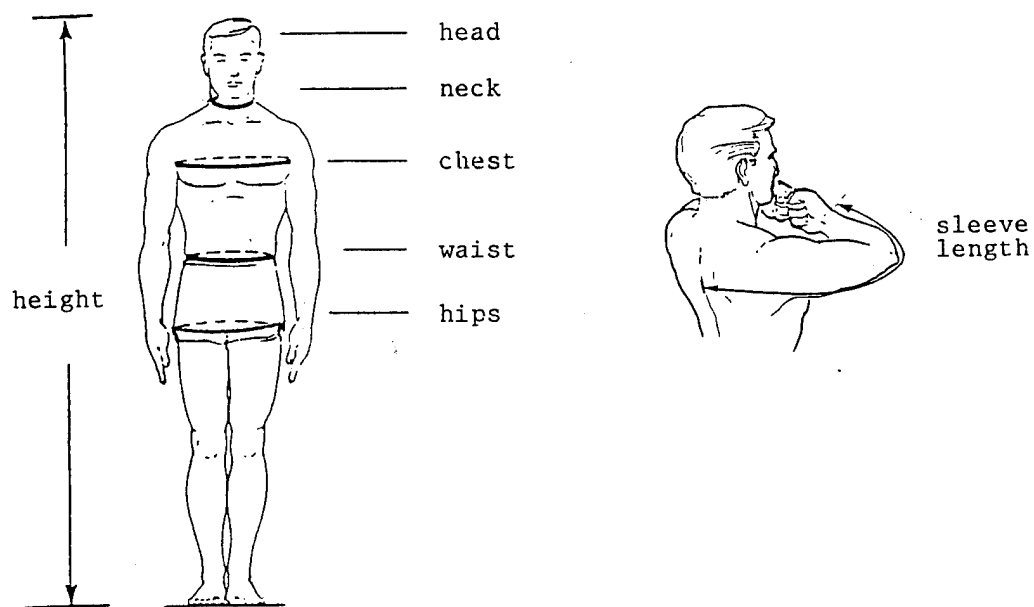


Figure 1. Measurements comprising a case.

length could be entered into separate fields. If in a single field, composite size would be an "outcome" field (the field to be predicted by the system), and therefore its domain should be ordered to facilitate indexing. We were unable to determine an appropriate coding for composite sizes, one that would establish a meaningful ordering among the size values. For example, one single-field coding option involves converting coat/trouser lengths to decimal values. e.g. 36S could be 36.0, 36R could be 36.2, etc. But this approach might cause a problem of interpretation, in that 35.9 would be interpreted as closer to 36.0 than would 36.2, for example, even though the former is a different numeric size.

To avoid possible problems with composite size, we put the numeric size and length into separate fields. This would allow designating the numeric size as the "outcome" field for the purpose of indexing, making it the field value to be predicted when the system is presented a new problem.

For composite-size garments, the initial plan was to predict length separately from numeric size, which would require two separate index structures for the cases, and separate retrievals to determine numeric size and length. (To establish an index when length is the outcome would require defining for the CBR shell an ordering of the symbolic lengths: XS<S<R<L<XL, which is easily done in ReMind.) Predicting length separately might be necessary if numeric size and length depend on entirely different factors, but the advice of expert fitters and results of experimenting with alternative strategies suggested that the numeric size and the length depend on similar factors. In other words, when the system uses the index based on the numeric size outcome to retrieve a case having a set of body measurements very similar to the new case, both the numeric size and the length from the retrieved case are likely to be correct for the new case. Therefore we decided to index using just the numeric size as the outcome field, and when predicting to select both the numeric size and the length from the most similar retrieved cases.

To speed up retrieval, for each garment type an index hierarchy (search tree) was developed automatically by the CBR shell. Beginning at the root of the search tree, this process selected at each node the value of the field which is the best discriminator, meaning that two branches can be created, representing roughly equal-sized sub-trees of cases, with the outcomes of cases in one sub-tree greater than those in the other. (Since little domain knowledge was available, the project team did not attempt to guide the indexing process with manual specifications.) Development of the search tree terminated when the number of cases in the leaf nodes dropped below a user-specified threshold for splitting. The threshold should be low enough such that relatively few leaf nodes contain cases with mixed outcomes having no clear majority, but there is no need to develop the tree to the extent that the lower branches are meaningless. For this project a reasonable choice was to develop the search tree such that leaf nodes would not be split if they contained less than ten cases. Figure 2 shows part of the index hierarchy for the short sleeve shirt. Each leaf of the tree represents a cluster of cases and is labelled with the number of cases having a particular outcome. For example, the upper cluster at the far right of the figure indicates two cases of size 15.5 and four cases of size 15.

This system is supposed to be a problem solver. Its purpose is to predict garment sizes for individuals. For each garment it should recommend a size (or a prioritized set of sizes) rather than just retrieving similar cases whose sizes might not all be the same, leaving it to the user to decide which is appropriate. Determining recommended size(s) was accomplished in three steps. First, a search using the index hierarchy located a minimum of 20 cases. (A reasonable number of cases should be retrieved to help compensate for the tendency for branches near the bottom of the tree to be meaningless). The remaining steps were necessary to distinguish among possibly mixed outcomes among the retrieved cases. (For example, some of the retrieved cases might have short sleeve shirt size 15.5 and others 15.0.)

The next step was to calculate similarity scores for the retrieved cases, using a routine built-in to the shell. That routine considers user-assigned weights for fields which varies their importance in determining similarity of cases. The system looks at every case in the database and normalizes values in each of the fields by assigning them a number in the range 0 through 100. Then, for each field, the system calculates a value representing the "difference" between the field value for the current case and the field value for the retrieved case. Finally, a formula accounts for the difference in values of all the fields and the user-assigned field weights to determine an overall similarity score ranging from 0 to 100 (the formula depends on the data type of the fields). Since our domain knowledge was limited we experimented with various weight assignments to help select appropriate weights for calculation of similarities. For example, the initial choice of weights for predicting size of the short sleeve shirt followed from the intuition that neck size was most important: neck 16, chest 8, weight 2, height 2, and other fields 0. Performance of this strategy was not good (the testing approach is described later). The best alternative we tried was to assign field weights in accordance with priorities which had been established in the automatic generation of the index hierarchy: weight 16, neck 8, chest 4, height 2, and other fields 0.

The third and final step was to take a vote among the 10 cases with the highest similarity scores. If the vote was not unanimous, the system reported its first, second and third recommendations for garment size. The second and third choices could be helpful at the clothing issue facility, because if the first choice garment is not appropriate the fitter would have recommendations for selecting other sizes for try-on.

Developing the User Interface

All the aforementioned features could be implemented in a straightforward way with the interactive version of the ReMind

shell, but the user would have to know how to use the shell and would have to take many point-and-click actions to accomplish size prediction for a single individual. Even if the user were very familiar with the shell, the process would be too tedious to be suitable in an operational environment. It was evident that predictions should be handled by a compiled program. Therefore, we developed a windows-based application using C++ and the run-time libraries furnished with the ReMind shell. The system accepts body measurements for an individual and prints 1st, 2nd and 3rd recommendations for sizes for each garment. Its major components are shown in Figure 3. The user interface accepts and validates input and provides results. The knowledge base contains cases consisting of body measurements together with garment sizes. The prediction engine follows the procedure outlined in the previous section (Figure 4).

Learning and Heuristics

Under normal circumstances this particular system has enough stored cases that it would not need to "learn" by adding new cases. However it would be wise to add a new case if it represents a body which is not well represented in the database of cases. The criteria for "well represented" could be one like this: there should exist at least x cases in the database all with a similarity score of at least y . If the criteria is not satisfied, the case should be added, thus increasing the chance that a future soldier with a similar body will be well represented. Also, it would be appropriate to add cases or totally replace cases if there is a significant change in garment dimensions, fitting policy, or soldier population. For example, there was a recent change in manufacturing of the Army green coat which resulted in different dimensions in the chest area. Fitting policy could vary in terms of how snugly uniforms should fit. During mobilization military services might expand their recruiting to include older enlistees who generally have different body dimensions than younger recruits.

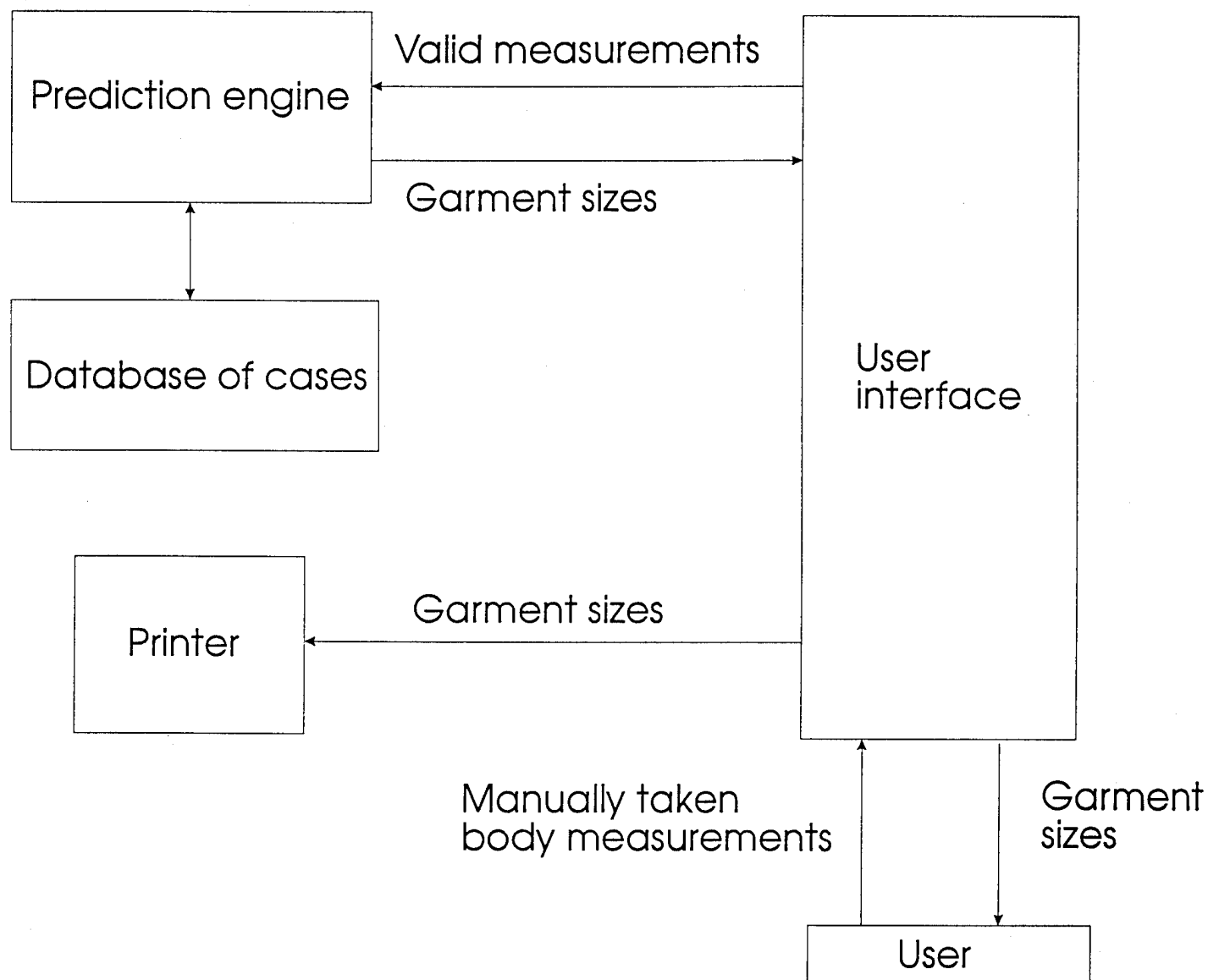


Figure 3. Major components of the system.

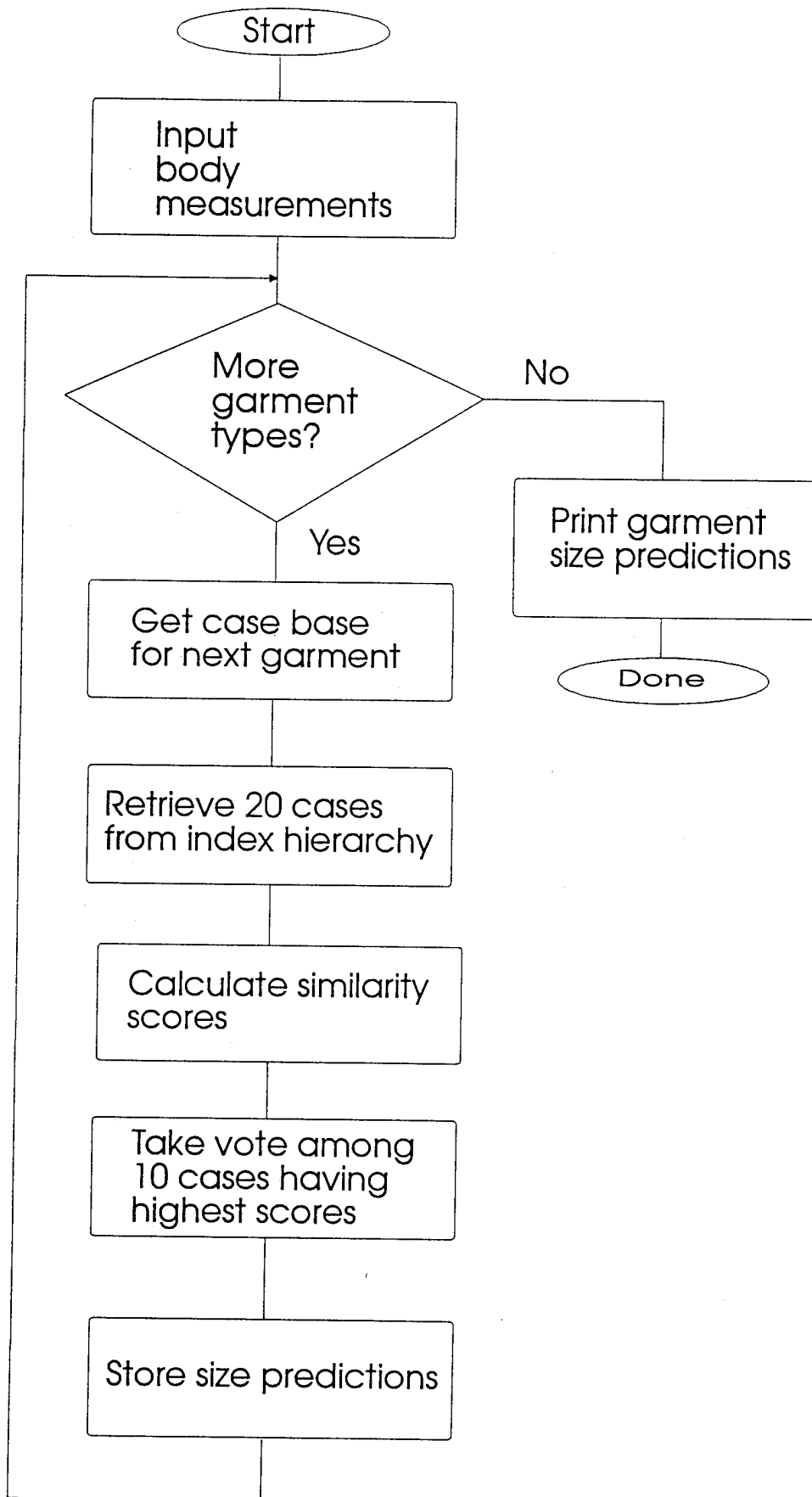


Figure 4. Procedure for predicting sizes.

Although they are often valuable in case-based systems, heuristics appeared unnecessary in this system. Here is a hypothetical situation where heuristics could be appropriate. Suppose the system is in operational use and is confronted with a soldier whose body measurements are so unusual that there are no similar cases in the database (what is "similar" enough is a judgement call of the system designer; the decision would be based on the similarity scores of the retrieved cases). Then one could employ a heuristic to infer an appropriate size. For example suppose the case in the database most similar to the soldier being fitted has the same dimensions in everything except height. The soldier's height is 3" more than that of the retrieved case. It may be reasonable to have a rule like the following in the system which is invoked in such instances:

IF the difference in all dimensions except height
 is less than 1 inch AND
 the difference in height is in range [2,4] inches
THEN the recommended size of coat
 is the size in the retrieved case AND
 the recommended length of coat
 is the next size longer than the length
 in the retrieved case

Heuristics were not needed in our system because 1) there are a very small percentage of soldiers with body measurements so unusual that they would not match up with cases in the database, and 2) usually such soldiers would need specially tailored garments anyway, so the systems's size prediction is not important.

Evaluation

There were two important questions: 1) how accurate are the system's predictions, and 2) can the system predict fast enough to be useful in practice?

To determine system accuracy we randomly selected 100 soldiers who had been issued garments at Fort Jackson, SC and collected their clothing records. Since body measurements and garment sizes for these soldiers were known, the data was suitable for testing. The system could be evaluated on the percentage of correct predictions for those soldiers. The system performed rather well, about the same as human experts. For example, the first choices for short sleeve shirt size were correct for 67% of the test cases, the first and second choices included the correct size for 94% of them. Even when incorrect, the predictions were almost always within a half size of the right one. If it were faster than human experts, a system with this level of accuracy could significantly streamline clothing issue operations at military facilities.

To evaluate system speed and practicality in an operational environment we conducted operational tests on two different dates at the clothing issue facility at Fort Jackson, SC. The tests were successful. We set up a personal computer system in the room where soldiers were measured. Right after being measured, each soldier called out the body measurements which had been entered on the clothing form and an operator entered the data into the computer. The computer printed garment size predictions on a page which was placed on a clipboard carried by the soldier throughout the fitting process. Fitters consulted that page to select garments for try on, rather than estimating sizes themselves.

The data input, calculation and output required only about 1 minute per soldier, which was fast enough not to delay the normal measuring process. Over 400 soldiers were processed in the two tests. The only delay occurred during a half hour period in the second test when there were temporarily two measurers working in parallel; normally there is just one. This generated a small backlog of soldiers waiting to be handled by the size prediction system. Other than that brief period, the size prediction system fit smoothly into the current operation. It was obvious that the system works faster than human experts. Each soldier is

served by a different fitter for each of the five garments. Without the size prediction system, each fitter takes a few seconds to examine the soldier, look at the measurements on his clothing record, and decide on a recommended size. With the prediction system those actions are unnecessary, because the recommended sizes are already available. Since the system introduces no significant delay to get input data and saves time at each of the five fitting stations, we concluded that the system is practical to employ in an operational environment. Before recommending the system for permanent installation, the authors wish to integrate a 3D body scanner.

Preparation to Incorporate a 3D Scanner

Although it predicts sizes rather well, the current prototype system is based entirely on manual measurements whose accuracy is questionable. Inaccuracies generally result from improper placement of the measuring tape and from variations in its tension. Even an experienced fitter's measurements are inconsistent because the fitter gets tired. Accuracy of the prediction system could be improved by integrating an automated measuring device, for example one based on a 3D scanner.

Practical 3D body scanners should be available in the near future. Three U.S. companies already have functional prototypes. Cyberware of Monterey, California, and Laser Design in Minneapolis are developing laser-based systems. Textile/Clothing Technology Corporation in Cary, N.C. is developing a white light system. Although they employ different technologies, all the scanners produce a set of three dimensional points (with x, y and z values).

It would be natural and convenient to employ the entire, unprocessed 3D body image directly in a CBR system. Although conceptually pleasing, that approach appears impractical in the near future. Apparently little is known about how to index complex graphical data [3], which is necessary if such data is to be effectively used in a CBR system. The next best thing is to

extract a few specific features (body measurements) from the 3D image and to use those features to define cases.

We decided to incorporate a 3D scanner in the size prediction system in two phases. In the first phase we would develop computer programs to make the scanner output useful to a case-based system by converting the set of 3D points to conventional body measurements (such as waist and chest). Although we would not have an actual scanner, we could with confidence develop an interface between it and a case-based system, knowing the scanner would provide 3D points. In the second phase, we would experiment with conventional and unconventional measurements to find a set which is effective in size prediction. In both phases, the calculated measurements would be used to compare the soldier being fitted to the cases in the database. Phase 1 is complete, and Phase 2 is awaiting acquisition of a 3D scanner.

Determining how body measurements should be calculated was challenging at first. It appeared necessary to resolve questions such as: given just a set of 3D points, what constitutes the chest or the waist measurement? Even if one decides those measures shall be horizontal circumferences, at what vertical point shall they be defined? It is necessary to determine precise definitions for those measures, some of which currently might be a matter of human judgement. (E.g. for some the waist is the largest and for others it is the smallest circumference of horizontal cross sections in the torso.) Fortunately, we later realized that for the purpose of automatic size prediction it is not necessary to take measures exactly as a human expert would. The important thing is to be consistent, which is no problem for a computer program.

It appeared wise to develop a flexible system for calculating measurements, since it might be desirable to experiment with various definitions of body measurements to seek those which are most effective in predicting sizes. To achieve flexibility we developed a command language which provides a convenient basis for calculating measurements. There are commands to isolate regions of the body such as shoulders, torso, and right leg.

Other commands tell the computer to calculate circumference at specific places, to find the minimum/maximum circumference, or to find certain surface points. These commands can be used to calculate measures automatically, including the traditional ones. For example, the chest might be measured with the following sequence of commands: go to the top of the torso, move 10% of the distance down, calculate the horizontal circumference. A graphical representation of this measurement on a 3D rendering of the body surface is shown in Figure 5.

Future Research and Development

Generally, performance of a case-based-reasoning system improves as the number of cases is increased, because the chance increases of retrieving a case which is very similar to a new problem. But at some point the rate of improvement tends to level off, because many of the newly added cases are the same as, or very similar to, cases already in the system. Knowing the "learning curve" can be helpful to a system developer. Certainly cases should be added as long as they tend to improve system performance, but when the marginal benefit of adding cases becomes small, continuing to add them could be a wasted effort and could adversely affect the speed of retrieval. (In this project, indexing of 2500 cases took about an hour on an IBM PS/2 with 386 processor.) To investigate the "learning curve" of the system we plan to measure system performance as cases are added.

Once the 3D scanner is integrated, experiments will be conducted to determine which measurements are most powerful in predicting garment sizes. It might turn out that non-traditional measures are most useful in predicting sizes, and some of them may be easy to get from the 3-D points. For example, it would be easy to compute the circumference of horizontal cross sections taken at regular intervals. As long as the measurements are effective in distinguishing body shapes, they do not have to be meaningful in the manual fitting process.

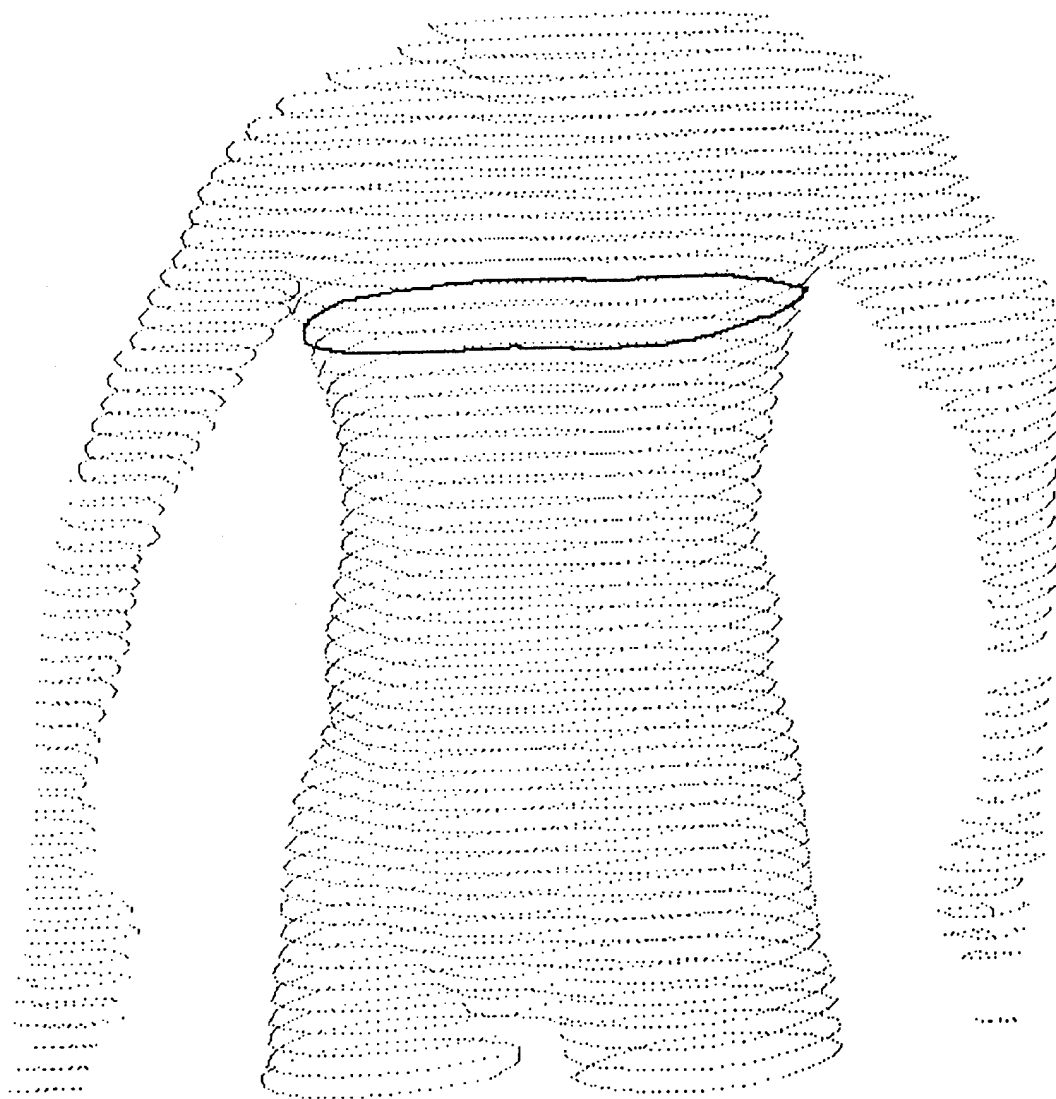


Figure 5. Chest measurement on 3D body image

When a 3D scanner is integrated, the system could be extended to facilitate made-to-measure manufacturing. The objective would be to electronically provide body dimensions as input to one of the current software packages for pattern alteration. That package can then generate a custom pattern for cutting the cloth. This capability would expedite the military's handling of soldiers who cannot be fitted with standard sizes.

Concluding Remarks

CBR appears to have been an appropriate choice for this system, even though our initial inclination was to employ the rule-based approach with which we were more familiar. CBR made the system easier to build. Cases were readily available, but it would have been difficult to acquire rules from the expert fitters.

CBR provides a better way to explain system reasoning than we could have achieved with a rule-based system. A CBR system can explain results by showing examples of previous real cases that are similar to the problem at hand. Rule-based systems can only report to the user the chain of rules that led to a conclusion. A rule-based system does not normally present examples or cases (even though its rules may be based on cases). In this project, the explanation capability will probably be important when the system is first installed, to help users develop confidence in its predictions. Afterward, this capability would probably be used only by the developers to investigate any problems in system operation.

Many other applications might benefit from employing 3D scanning with case-based reasoning. For example a case-based system has been built for process planning, wherein cases consist of a part description together with an appropriate process for manufacture [4]. To select a plan for a new part, one performs a geometrical comparison with that part with those in the database. That comparison is based on a few manually measured features. Perhaps the system effectiveness could be improved with a more thorough comparison based on a 3D scan.

Acknowledgements

The Defense Logistics Agency supported this project under Contract DLA 900-87-D-0017, DO 0026. Don O'Brien and Julie Tsao of the DLA gave advice and encouragement. Cognitive Systems, Inc. provided software support. Chris Jarvis and Jack Peck, Co-Directors of Clemson Apparel Research, provided direction and assistance. Contributors included Sarat Vemuri (CBR programming), Murali Earagolla (developing cases), Shan Jiang, Jasbir Manotra, and Prahlad Yerra (graphics programming). Janet Kolodner provided technical advice.

References

1. R. Creecy, B. Masand, S. Smith and D. Waltz, "Trading MIPS and Memory for Knowledge Engineering," Communications of the ACM, Vol. 35, No. 8, 1992, pp. 48-63.
2. J. Kolodner and W. Mark, "Case-Based Reasoning," IEEE Expert, Vol. 7, No. 5, Oct. 1992, pp. 5-6.
3. S. Chang and A. Hsu, "Image Information systems: Where Do We Go From Here," IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 5, Oct. 1992, pp. 431-442.
4. B. Humm, C.H. Schulz, M. Radtke and G. Warnecke, "A System for Case-based Process Planning," Computers in Industry 17, Vol. 17, 1991, pp. 169-180.

Appendix I

Newspaper article, Greenville News, June 3, 1993

THURSDAY, JUNE 3, 1993

CLASSIFIED • LOCAL • OBITUARIES

SECTION

C

THE NEWS

PICKENS/O'CONNOR

Clemson device to fit suits laser sharp

By Ron Barnett
News-Clemson Bureau

CLEMSON — That well-dressed gentleman with a snappy yellow tape measure who sizes you up for a suit could be replaced by a laser device that can take your measurements without ever touching you, Clemson University officials said.

The Clemson Apparel Research Center, under contract with the U.S. Army's Defense Lo-

gistics Agency, is involved in developing applications of a device a little bigger than a phone booth that could transform the apparel industry, officials said.

The Clemson University Research Foundation on Wednesday approved borrowing up to \$300,000 to buy a prototype of the device, called a 3-D non-contact body scan, assuming the contract with the Army is extended to recoup cost of the device.

Three companies are working

toward developing such devices, and Clemson would be the first to put it to use, sizing up soldiers at Fort Jackson for uniforms, a researcher on the project said.

"Three years ago when I first started talking about it, I was 'that crazy lady at Clemson,'" said Nancy Staples, principal investigator on the project.

"And by this year, everybody's talking about doing something like this."

A prototype that uses white

light instead of laser is due to arrive at Clemson next month, she said.

Two other systems that use low-powered lasers also will be considered before a decision on which one to buy is made, Ms. Staples said.

The project also includes development of an "expert system," or computer program that will attempt to simulate the expertise of the suit salesman with an eye for what size suit would work best for

an individual, she said.

Scanning devices like the one under development for Clemson already are being used for certain medical procedures and for such things as making masks used in Hollywood productions, she said.

But one big enough to measure a whole body has never been made, she said.

"In effect, you will walk into a box, an enclosed space, and stand in a position that you're told to stand," she said.

Computer coordinates of the body dimensions can be generated as quick as 1/30 of a second or up to 17 seconds depending on which technology is used, she said.

Beyond the military application, the device, coupled with the computer program for recommending an expert fit, could make custom-fitted clothing much more common in the future, she said.

Appendix J

Published trade article, AIM August 1994

TECHNOLOGY

3-D Body Scanning Gets High Marks

Female respondents say they would be willing to pay up to \$10 for body scans in order to get clothes that fit. □ Excerpted from a survey report by Audra Knight and Nancy Cassill, Ph.D.

About half of the respondents in this consumer survey on body scanning say 3-D scanning is a useful way to get a good fit in clothing, and most say they would be willing to pay up to \$10 for it. The study results indicate that apparel manufacturers and retailers should consider purchasing body-scanning devices for in-store use.

The survey on 3-D scanning, entitled "The Market Feasibility of Body Scanning and Size Prediction Technologies at Retail," was conducted by Audra

Knight, who recently received her master's degree from the School of Human Environmental Sciences, Department of Clothing and Textiles, at the University of North Carolina at Greensboro, under the guidance of Dr. Nancy Cassill, an associate professor. The purpose of the study was to determine the market feasibility of a body scanning system for customers in retail stores.

"This survey took a marketing orientation," says Cassill. "Rather than developing technology and then trying to sell it to the consumer, this study asks consumers

what they want so the appropriate technology can be developed for their use." She says she has discussed the study with representatives of several VF Corp. subsidiaries, but at press time none of these companies had committed to using the technology.

Fitting individuals. In a report on the study results, Knight writes that consumers need apparel fitted on a 3-D human body. "Historically, ready-to-wear apparel has been created by designers preparing 2-D 'design concepts' that treat the human body as if it were flat and divided into separate unconnected segments. The resulting products are designed

- 
- ▲ 76% of respondents found body scan appealing
 - ▲ size, age had no effect on attitude toward scanning
 - ▲ would use scan to buy bottoms, swimwear, tops

Appeal of Body Scanning

	Not at all	Somewhat	Very
(a) Made-to-measure	25.8%	27.8%	46.0%
(b) Data card	20.6%	19.6%	58.7%
(c) Computer imaging	26.8%	17.5%	54.0%

Note: Percentages based on sample size of 97.

for the general mass market, not individuals," she says. Knight adds that sizing standards are based on data collected decades ago that are no longer applicable to current needs. She notes the latest generation of computer-aided design systems makes it possible to create garment shapes that are 3-D and incorporate precise human measurements. She adds that other industries, such as the footwear industry, have developed similar scanning systems and have used them for years to ensure proper size and fit.

Knight says that before she conducted her study, there had been no consumer research on the market feasibility of body scanning. The random sample group chosen for the study consisted of 200 female employees at UNC-G, and a five-page questionnaire containing questions on the acceptability of body scanning, sizing, fit problems and demographics was mailed to the group in February 1994. The questionnaire also contained a description of a body scan procedure.

According to the description, a computerized booth would be located in the dressing room area of a retail store. The booth would be black with dim lighting and operated by the consumer, who would remove all clothing except underwear and push a button to take exact body measurements. The

process would take six seconds. After measurements are taken, they could be transferred to an electronic card or kept in a data base. An electronic card could be produced in a few minutes for the customer. If measurements are kept in a database, they would be available to the customer at any store with the scanning technology. Body information would be updated as often as a customer chooses. A total of 97 women returned the questionnaire, for a 48.5% response rate.

According to the study, women are the primary purchasers of apparel. Fifty-five percent of all women in

Profile of Survey Respondents

The women in the group were employed at UNC-G in positions ranging from research assistant to secretarial to housekeeping. The largest group of respondents, 36.1%, were 30 to 39 years of age; 16.5% were under age 30; 25.8% ages 40-49; and 19.6% 50 and above. The annual household income level of 24.7% of the respondents was below \$30,000; 31% had an income of \$30,000 to \$49,999; and 40.2%, above \$50,000. The majority of the sample, 84.5% was white, 11.3%

black, and 1% Asian (3.2% no response).

About half of the women had a high school diploma and some college or vocational training; 25.5% held an undergraduate degree and 10.3% had some graduate school. A total of 45.5% of women in the study spent between \$200 and \$499 on their wardrobe last year; and 30% spent between \$500 and \$999. Only 10.3% spent less than \$200; 11.3% spent more than \$2,000. Nearly all respondents, 97%, used a computer at their jobs. □

TECHNOLOGY

Scanning Technology Ensures Perfect Fit

New 3-D scanning technology is being developed by Textile/Clothing Technology Corp. and the Clemson Apparel Research Center, which is in the final stages of development and is expected to be in the marketplace by early 1995. The technology, known as the body scanner, will scan the body with a new non-contact measurement system.

Joseph Off, managing director of (TC), says body scanning technology probably will be ready for expanded testing early next year. (TC) has been working to develop scanning technology for more than three years.

"When it is final, we will test it and determine how good it is and make final corrections," he says, adding that (TC) also will begin to develop commercial markets for the technology.

CAR has developed software for scanning devices that will be used to read body measurements for made-to-measure apparel and to predict sizes for ready-to-wear clothing, says Nancy Shipley, research associate and assistant professor at CAR. She says CAR is waiting to conduct mass testing of the software program until scanning devices are fully developed. She adds that a defense department project to simplify uniform sizing has been put on hold until scanners are available.

When they are available, the scanners will be designed to take accurate measurements of the body in a computerized booth in six seconds or less. The body scan then can be used to manufacture made-to-measure clothing, generate a "data card" with all the consumer's body measurements and superimpose garments on the computer image of the consumer's body. □

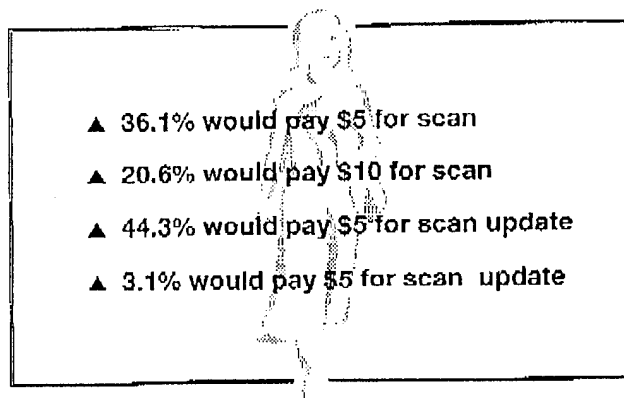
(TC), Cyberware, and Laser Design have been working to develop scanning techniques for many years.

She adds that research to simplify uniform issue has been put on hold until scanners are available.

less than
20 sec

the United States are in the labor force; 65% of women ages 18 to 34 are employed, 70% of women 35 to 54 are employed, and 20% of women 55 and older are employed. Middle-aged women are 20% more likely to buy a suit each year, and they are 40% more likely to buy a blazer. Sixty percent of them buy four or more dresses a year, compared with 40% of younger women and fewer than 30% of older women.

The study also shows that employed women place greater value on time-saving, convenience-shopping outlets, place greater emphasis on fashion, take considerable interest in clothing's flattering qualities and spend more on clothing. Working women are more than twice as likely to spend \$500 or more on apparel annually than non-employed women, the study shows.



Aceptability of body scanning. The first objective of the study was to determine consumer interest in body scanning, and more specifically, the acceptability of three body scan methods: made to measure, data card and computer imaging. Seventy-six percent of all of the participants were willing to have a body scan. The data card was the most appealing method, followed by computer imaging. The made-to-measure method, in which measurements are taken for an individual but not used to determine industry-wide sizing, was the least appealing, according to participants' responses. Some respondents, 36.5% to 44.8%, found the applications more appealing than usable; about half, 50% to 54.2%, found them equally appealing and usable; and 5.2% to 9.3% found the applications more usable than appealing.

Study participants said scanning technology should be located in each dressing room area. Less than half said they would be more willing to use body-scanning techniques if they could wear a body

TECHNOLOGY

Respondents' Annual Spending on Apparel			
less than \$200	\$200 - \$499	\$500 - \$999	\$1,000+
10.3%	45.4%	30.0%	11.3%
No response: 3.0%			

suit. The women said they would use body scanning when purchasing jeans and slacks, swimwear and blouses and shirts. A total of 36.1% said they would be willing to pay \$5 for an initial scan, and 20.6% said they would pay up to \$10. A total of 44.3% said they would pay \$5 each time they updated their card, and 3.1% said they would be willing to spend \$10 on card updates.

The second objective was to determine if participants have problems with the fit of everyday apparel and jeans. Almost half said they sometimes alter their clothing. In particular, the women said they have problems with the waists, hips and lengths of jeans and that it is common for them to try on more than one pair before finding a proper fit. More than half said they would be willing to pay more for apparel made to their own size specifications.

Neither the age nor body size of the consumer affects the appeal or usage of the three body scanning techniques, according to the survey. A greater number of small-sized respondents most often would use the made-to-measure method, and more large-sized consumers would use computer imaging, the survey showed.

"The main objections to body scanning were expressed by participants who had no problems with fit and believed scanning would be a waste of time for them," Knight says. "Some of the participants thought that body scanning would be just another time-consuming step in shopping, that they would not save any time over the traditional trying-on process."

Implications for retailers, manufacturers. As a result of the responses, the survey recommends that retailers and manufacturers further explore the appeal and use of body scanning technologies and that they consider purchasing the devices for retail stores. Noting that there are about 135,000 retailers and 21,301 apparel or textile product companies in the United States, Knight says in the survey report, "Retailers and manufacturers have realized that developing strong, mutually interdependent relationships is critical in achieving consumer satisfaction. The heart of retailer-manufacturer relationships is the sharing of information through the use of technology."

Usability of Body Scanning

	Not at all	Somewhat	Very
(a) Made-to-measure	42.2%	29.9%	26.8%
(b) Data card	32.0%	17.5%	49.5%
(c) Computer imaging	35.1%	20.6%	43.2%

Note: Percentages based on sample size of 97.

The study also recommends further study of demographic variables, such as income and race, that may affect consumer opinion and selection of a niche market such as athletic or fitness to compare fit problems. □

Audra Knight recently received her master's degree from the School of Human Environmental Sciences, Department of Clothing and Textiles, at the University of North Carolina at Greensboro. Dr. Nancy Cassill is an associate professor there.

Appendix K

Published trade article, AIM October 1994

Body Scanning In The Future

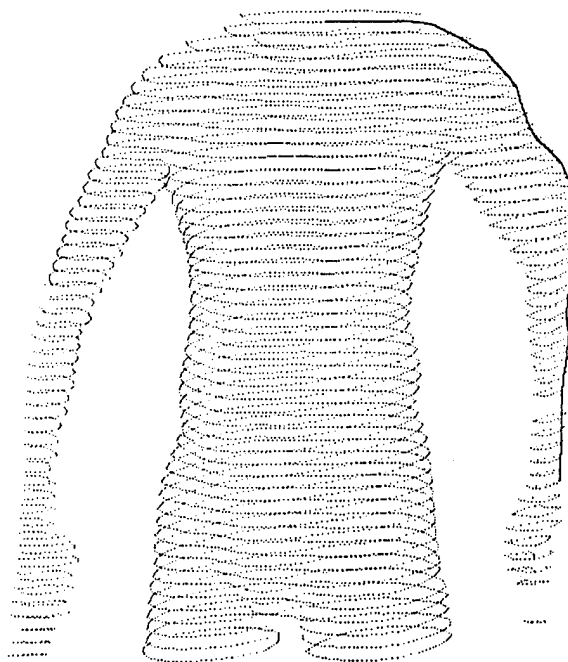
With the technology to effectively use 3-D body scans just around the corner, apparel manufacturers need to make plans. — by Nancy Staples, Ph.D., Roy Pargas, Ph.D. and Steve Davis, Ph.D.

Picture this: In the not-so-distant future, you will be measured for clothing with 3-D, non-contact body scanners. You will step into a booth and through the magic of technology, the contours of your body will be reproduced on a computer screen. The attendant will hand you a diskette containing your data: a file of x, y and z points that, with the proper software on a computer of a sufficient size, can be converted to the display on the computer screen. What? You thought you were going to get your body measured and instantly get a better fit in your clothes as a result?

The fact is that the body scan is only the beginning. (See "3-D Body Scanning Gets High Marks," *AIM*, August 1994, page 98.) The output is useless without the software to extract information, such as measurements, from the scan. These body dimensions then can be used by anthropologists to create more accurate anthropometric data bases that can be analyzed for designing people-friendly work spaces; by apparel manufacturers to develop better fit models and size ranges of stock clothing sizes or to make made-to-measure or custom-made clothing more affordable; and by health professionals to study growth in children or to analyze body changes due to dieting or exercise.

Be prepared. What's next? The industry must plan ahead for the inevitability of body scanning technology.

When body scanners are ready to be field tested, the people who plan to use them must be prepared with a knowledge of precisely what they need from the output of the scan. In terms of measurements, this means a clear definition of the body dimensions desired and their precise locations. If manufacturers and retailers want to know more about the size and shape of their



This illustration shows the highlighted surface distance to be measured (before smoothing).

customers, they will need to determine what data will be most useful. This will depend on using the data for model customization for the target customer; style and block pattern development; or computer-aided design alterations to modify existing patterns. If made-to-measure becomes more simple to accomplish at a comparable cost to stock sizes, then there must be manufacturers flexible enough to be able to produce customized garments with little or no disruption of their work flow. If a retailer wants to assist customers in selecting a best-fit size, then the dimensions or shapes that best predict sizes for their products must be determined.

Continued on Page 50

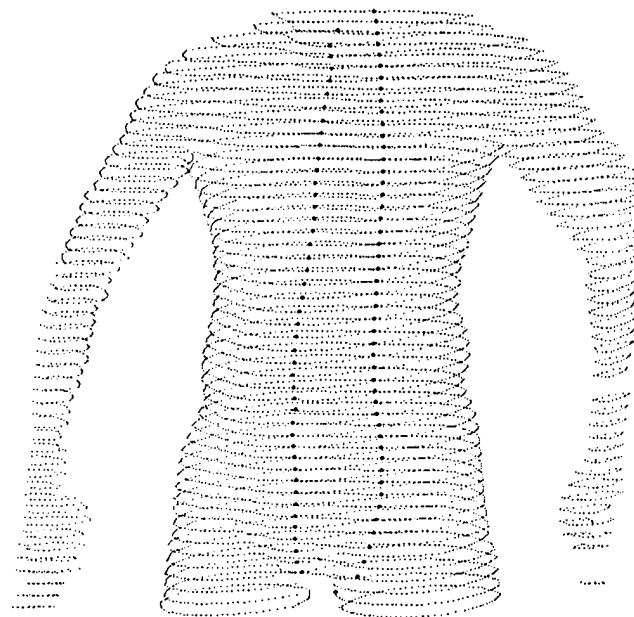
Other issues must be resolved, such as who owns the data. Does the consumer carry around his or her scan data on an electronic storage device or is it stored in a central data base? If it is stored in a central data base, then who has access to the data?

The prospect of 3-D body scanning becoming a reality in the near future is exciting to researchers and applications people alike. We will have access to information that, if correctly analyzed and applied, has the potential to transform the apparel industry and the quality of the products it produces.

Currently, three U.S. companies are the closest to producing a field-test-worthy full body scanner. Cyberware of Monterey, Calif., and Laser Design in Minneapolis are developing laser-based systems. Textile/Clothing Technology Corp. in Cary, N.C., is developing a white-light system. The earliest expected functional prototype demonstration was expected this month, with the earliest expected delivery of a scanner in the first quarter of 1995. The U.S. Air Force and Army will be the first customers.

How it is done. Data from a body scan can be expressed as x, y and z points and displayed on a computer screen or further manipulated using existing data editing software to simulate the addition of skin.

In 1992, the Defense Logistics Agency, a branch of the United States Defense Department responsible for procuring uniforms, funded a research project at Clemson Apparel Research to create the software for determining body dimensions from a 3-D body scan and for



Vertical slice selected.

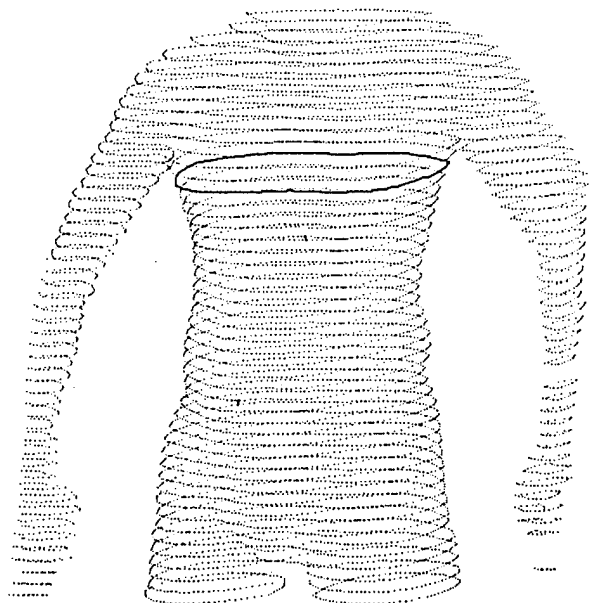
using those dimensions to predict uniform sizes for initial dress uniform issue. Because the Army currently records the chest, waist, seat, neck and head circumferences and the sleeve length of its recruits, work initially focused on duplicating the manual process and deriving these same measurements from a body scan.

The first tool developed was for determining the circumference of horizontal slices. When the operator points with a mouse and clicks on a point at the level where the measurement needs to be determined, all of the points at that same "y" value are highlighted. The computer then displays the circumference of the slice in the measurement box.

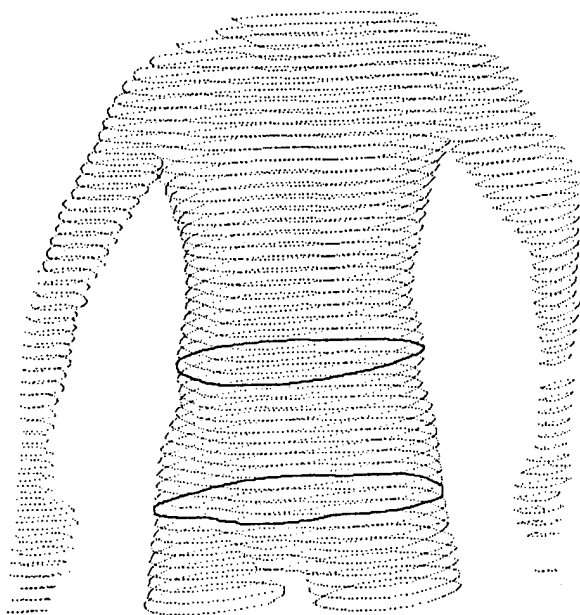
The next tool needed was for surface distance measurements (as in sleeve length). The operator can click on a series of points, such as the center back at neck (P1), the shoulder tip (P2), the elbow tip (P3) and the side of the wrist (P4), and the computer calculates the shortest distance from P1 to P4 on the surface of the body. The resulting line looks a bit lumpy at first, but after smoothing, simulates the equivalent of a tape measure being placed on the body.

With these two tools, the Army's standard measurements could be derived from a body scan, but the interactive nature of the software was cumbersome and time consuming. The next step was to attempt to automate the selection of the measurement location.

Tools were created to isolate the regions of the body such as shoulders, torso, right leg, left leg, right arm and left arm. It then was possible to tell the computer to select a slice from 0% to 100% of the distance from the top of the region to the bottom, move down a give-



Highlighted multiple slices.



Highlighted slice for circumference measurement.

number of inches from that slice and take a circumference measurement (example: slice 0% of the torso, move 2", measure, call the result "chest"). It also is possible to tell the computer to find the largest or smallest circumference in the region (the waist in the female is the minimum circumference in the torso region).

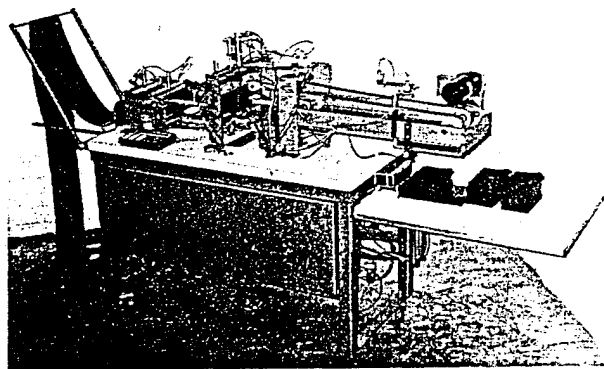
To make the selection of surface points more precise, tools were created to select the center point on front, back, left or right of a slice or the "most" point (most right, most left). By selecting the correct series of slices and selecting the appropriate "center" of most points on each, the sleeve length definition becomes more precise (P1 equals slice 0% of the shoulder region at the base of the neck, center back point; P2 equals slice 100% of the shoulder region at the base of the shoulder, left most point; P3 equals slice with left most point of the scan at the elbow, left most point; and P4 equals slice with the minimum circumference on the left arm at the wrist, left most point).

The commands to perform the selection of regions, slices and points compose the building blocks of a language that can be combined to custom design the soft-

ware to
who ar
ing bo
uring
scan w
ments
of bod
resulti
dimen
cation
it wor
measu
sired.

Var
crease
can be
two c
displa
be es
wear
width
wide
Ve

SAVE TIME AND MONEY WITH THE NEW KEETON TURTLE NECK CUT AND TURN MACHINE



- Cuts & turns any size neck from a tube
- Fast (approximately 1 min.) loading time
- Required size easily punched in
- Turns, cuts & indexes your determined number per stack
- Cycle time approximately 4 seconds
- One operator easily runs two machines

For more information or to place your order, call:

Keeton Products, Inc.

P.O. Box 296 • Campbellsville, Ky. 42718

(502) 465-7433 • Fax: (502) 789-2371

Circle No. 33 on Reader Service Card

ware to a user's needs. Even among anthropologists who are taught the most scientific method of measuring bodies, there is variation in the placement of measuring devices. Measurements derived from a body scan will be most useful if the locations of the measurements are clearly defined and accompany any listing of body dimensions. In this way, anyone looking at a resulting table of measurements will know where the dimensions were located on the body. If a different location was preferred, since body scans can be stored, it would be possible to request that an additional measurement be recorded, based on the location desired.

Variations on the basic tools were developed to increase the usefulness of the software. Multiple slices can be selected and compared. The operator can select two or more slices and view them on the body, then display them in a combined overhead view. This will be especially useful in the development of women's wear, where the abdomen often determines the widest width of a skirt front while the buttocks determine the widest width of a skirt back.

Vertical slices of the body can be selected and

viewed straight on to assist in the definition of posture for tailored wear. Additional software will be written to direct the computer to analyze the vertical slice and determine the posture type it represents.

Non-traditional measurements can be derived and analyzed. It may be that a comparison of the long axis and the short axis of the base-of-the-shoulder slice, in combination with the chest measurement, will aid in the prediction of what tailored jacket size a man should wear. Because of access to multiple slice information, women's wear manufacturers may choose to offer new size types to accommodate variations in the relationship between rib cage size and shape and breast size in women. □

Scan data files for this research were created by Dimensional Measurement Systems Inc., New York, and provided to CAR by (TC), current owner of the former DMS equipment.

Nancy Staples, Ph.D. is a research associate/assistant professor at CAR. Roy Pargas, Ph.D., is an associate professor of computer science at Clemson University. Steve Davis, Ph.D., is a professor of management and computer science at Clemson University.

SCOTT

Scott Tag & Label Co., Inc.

COMPUTER
HANG TAGS
COMPUTER LABELS
BARCODE EXPERTS
EDI INTERFACE

Your Source...

SCOTT TAG & LABEL COMPANY is a company dedicated to excellence. We have innovative and creative ideas to solve YOUR problems and YOUR identification needs.

WE SELL SOLUTIONS

Solutions to Bar Code problems and needs...

Solutions to EDI confusions and demands...

Solutions to your software needs...

Solutions to shipping system compliance...

Solutions to service problems and support.

WE PRIDE OURSELVES ON SERVICE...

We service what we sell!

CORPORATE OFFICE

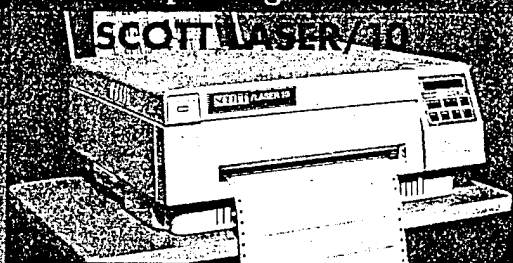
226 West 37th Street
New York, NY 10018
Tel: (212) 643-4100
Fax: (212) 643-8569
1-800-96 SCOTT

CALIFORNIA SALES OFFICE

141 Duesenberg Dr., Suite 15
Westlake Village, CA 91362
Tel: (805) 497-0964
Fax: (805) 497-3361

BARCODE AND EDI

Competitiveness in 1994 means meeting these demands efficiently and cost effectively. Scott Tag & Label Company introduces the most advanced method of in-plant barcode printing with the...



A continuous feed Laser Printing System for all varieties of Bar Codes

HIGH QUALITY: 90,000 pixels per square inch!

COST SAVINGS: Save 35% vs. thermal ribbon

LABOR SAVINGS: 90% less operator attendance

HIGH OUTPUT: Unmatched for daily production!

SUPPORT: Scott's own staff of programmers and technicians

The **SCOTT LASER/10** is available as a stand alone system or it can be integrated into your applications

CALL TODAY 1-800-96 SCOTT

SCOTT Scott Tag & Label Co., Inc.
is an associate member of the UCC EDI committee

Appendix L

Size prediction CDRs

SOFTWARE DEVELOPMENT PLAN—Size Prediction

1.0 *Introduction.*

This document establishes the software development plan for the size prediction system. This research and development project will explore modern approaches to developing an expert system. It will also seek to develop new algorithms for determining body measurements from a 3D body scan.

2.0 *Organization and Responsibility.*

This project is conducted by Nancy Staples, Roy Pargas, and Steve Davis as part of the Clemson Apparel Research project. Nancy Staples is the principal investigator.

3.0 *Management and Technical Controls.*

Project members will meet at least once a month to discuss progress and problems. Monthly reports to the sponsor will be used partly as an internal management tool to document progress and identify difficulties which arise.

4.0 *Resources.*

4.1 *Personnel.*

The following details the division of major responsibilities: Nancy Staples, project direction, development of schemes for measuring the body; Roy Pargas, development of algorithms and software for producing body measurements from the output of a 3D body scan (two part-time graduate students will assist Dr. Pargas); Steve Davis, development of an expert system which predicts garment sizes from body measurements (two part-time graduate students will assist Dr. Davis).

4.2 *Training.*

Davis and two graduate students should attend training in ReMind®, a case-based reasoning shell, no later than March 1993.

4.3 *Data Processing Equipment.*

This project will employ IBM-compatible personal computers which are already on hand at the Clemson Apparel Research facility.

5.0 *Software Development Schedule.*

The project work is organized in two main efforts: 3D body measurement and expert system. The 3D body measurement tasks are: study of 3D scanners, development of basic measures, development of measurement macro language, and complete measurement software. The expert system tasks are: training in ReMind®, gather soldier data, develop expert system, and test expert system.

Gantt Chart

Item	Quarter	1	2	3	4	5	6	7	8
Management Plan									
Study of 3D Scanners									
Development of basic measures									
Development of measurement macro language									
Complete measurement software									
Training in ReMind®									
Gather soldier data									
Develop expert system									
Test expert system									

6.0 Risk Areas.

There are two principal risk areas. First, the project is not assured of getting a 3D body scanner in a timely fashion. This technology is still under development. Second, the project team is employing a new software technology for the expert system, case-based reasoning. None of the project team has any experience with this approach or with tools which support it. To minimize the risk concerning the 3D scanner, the project will 1) search nationwide for potential developers of a scanner, and try to negotiate getting whichever product is first ready to be employed in this project; and 2) develop device-independent software for converting output of a 3D scanner into useful body measurements.

7.0 Monitoring and Reporting.

The primary scheme will be the monthly report to the project sponsor. This report will include status of program development, problems, risk areas, and planned solutions.

8.0 Documentation.

Since this project is exploratory, it will not attempt to develop documentation in a format appropriate for commercial software. Instead, the project will 1) devote portions of monthly reports to documenting software, and 2) employ reports prepared by programmers as part of their requirements for an academic program.

9.0 Development Approach.

The purpose of this project is to develop a prototype system, so the software will be a prototype also. The project will employ a case-based reasoning shell (ReMind®) and run-time libraries for the expert system.

10.0 *Use of Existing Software.*

10.1 *Commercially Developed Software.*

This project will use ReMind®, a case-based reasoning shell, to allow building the expert system faster. Documentation for ReMind® is furnished in a user manual. The project will have the rights to all data. There are no plans to certify ReMind®, since it already has a good reputation in the industry.

10.2 *Existing Applications Software.*

No software of this type will be used, with the exception of programs that may be furnished by the manufacturer of a 3D body scanner.

11.0 *Development and Test Tools.*

This project will develop a procedure for testing the expert system. The scheme calls for reserving a number of cases for testing, then comparing system predictions of garment sizes to the sizes associated with those cases.

12.0 *Security Controls and Requirements.*

Since results of this project are open to the public, no special security controls will be necessary.

SOFTWARE REQUIREMENTS SPECIFICATION—Size Prediction

1.0 *Introduction.*

This document establishes the requirements for the size prediction system.

1.1. *Purpose.*

The intent is to prescribe clearly how the system will perform.

1.2 *Scope.*

Requirements will be outlined.

1.3 *Terminology.*

3D Non-contact Measurement—refers to a machine which takes body measurements using light beams.

Case-based-reasoning (CBR)—describes an approach to developing expert systems which employs previous cases to determine the correct disposition of a current case.

CIIP—Clothing Initial Issue Point.

1.4 *References.* (none)

1.5 *Overview.*

2.0 *General description.*

The system will be developed in two phases. The Phase I system will take manual measurements as input and will predict the appropriate garment sizes. The Phase II system will employ 3D measurements as input.

2.1 *Product perspective.*

The Phase I system will consist of software running on an IBM-compatible PC located in or near the measurement room of a military clothing issue facility. The Phase II system will consist of a 3D non-contact measurement device connected to the PC system.

2.2 *Product functions.*

The system will predict soldier garment sizes, print measurements and sizes on either plain paper or on a portion of the clothing form. To facilitate evaluation it will store in a database: sizes and predictions, sizes actually assigned, sizes of garments tried on, and alteration data.

2.3 *User characteristics.*

Intended users are personnel currently conducting measurement and clothing issue activities at military installations.

2.4 *General constraints.*

The system must be usable by people without specialized training.

2.5 *Assumptions.* (n/a)

3.0 *Specific requirements*

3.1 *Functional requirements.*

3.1.1 *Size prediction.*

Phase I inputs (from the keyboard at the measurement room) include: soldier last name, first name, height, head, neck, chest, waist, hips, sleeve length, weight.

Phase II inputs (from 3D non-contact device) to be determined (it depends on the specific device).

Process: to predict sizes of short sleeve shirt, long sleeve shirt, black coat, green coat, and trousers.

Outputs: Last name, first name, size for each garment. In Phase II may also output alteration data.

3.1.2 *Evaluation of size prediction and update of cases.*

Inputs (at the issue point): sizes of garments tried on, and sizes assigned for each of the garments.

Process: storage in database; and calculation of average number of try-ons for each garment type; also storage of correct size assignment as a new case in the database.

Outputs: average number of try-ons.

3.2 *External interface requirements.*

3.2.1 *Human.*

Inputs in the measurement room will be accomplished by the measurer or his designated representative, who might be a soldier on detail. Outputs in the measurement room will be accomplished by a laser printer. If output is directed to the clothing form, the user (possibly the soldier involved) must enter the clothing form into the printer in the right way so the output will be directed to the appropriate part of the form.

Try-on data must be recorded on the clothing form by CIIP personnel (note this is a revision to normal procedure and requires their support). Sizes of garments tried on will be recorded next to the final size issued.

3.2.2 *Hardware.*

(see next paragraph; we might want to interface with the current mainframe-based system).

3.2.3 *Software.*

Consider merging the function of our system and the mainframe system currently employed at Ft. Jackson, SC. Current system involves keying data from the clothing form, which is later used to print a clothing form. Merging systems would save duplicate keying, and could convert from a mainframe to a PC-based system which might be more convenient.

3.2.4 *Communications.*

In a future version, we possibly may want to transmit electronically soldier names and sizes to clothing issue stations. In Phase II we may want to transmit

size and alteration requirements to the clothing issue point or alteration room.

3.2.5 Location.

The personal computer and printer will be located in the measurement room.

3.3.0 Performance requirements.

3.3.1 Input/output loads.

The system must handle 400 soldiers per day.

3.3.2 Database loads.

When the system is in use, it must retain daily an estimated 60K bytes (400 soldiers with 150 bytes per soldier record).

3.3.3 Response times.

The system must begin printing correct sizes within 20 seconds after entry of measurement input.

3.3.4 Resource usage.

Requires electrical power for computer system in measurement room.

3.4.0 Design constraints.

Although not a constraint, the current plan is to employ case-based reasoning (CBR) to enable the system to learn and to easily adapt to new situations (such as the Marine Corps issue facility).

3.4.1 Standards compliance.

Since this project will develop a research prototype, standards are not essential.

3.4.2 Hardware limitations.

System should run effectively on an IBM-compatible PC with Pentium processor.

3.5.0 Attributes (quality requirements).

The system should predict garment sizes at least as accurately as current human fitters. Their initial (first try) predictions are about 75% accurate for most garments.

3.5.1 Availability. (n/a; research prototype)

3.5.2 Reliability. (n/a; research prototype)

3.5.3 Maintainability. (n/a; research prototype)

SOFTWARE TEST DESCRIPTION—Size Prediction
(contract specifies a Software Systems Development Test and Evaluation Plan, DI-MCCR-8039; this outline is from DI-IPSC-81439, published in MIL-STD-498, 5 December 1994)

1.0 *Scope.*

1.1 *Identification.*

This describes software completed in contract DLA900-87-D-0017, D.O. 0026, which extracts body measurements from a 3D body scan and predicts clothing sizes from body measurements.

1.2 *System overview.*

Clemson Apparel Research began the project to design an expert system for initial try-on of the US Army men's dress uniform began on June 10, 1992. This was a contract sponsored by the Defense Logistics Agency. The objective was to automate the prediction of US Army male dress uniform initial issue try-on size by employing an expert system in coordination with accurate 3-dimensional, non-contact body measurement. The history of system development is described in detail in the final report of which this is a part. The Clothing Initial Issue Point (CIIP) at Fort Jackson, SC supported this project by serving as a test site.

1.3 *Document overview.*

This describes how the size prediction system may be tested.

2. *Referenced documents.*

- a. Jindal, V. "Investigation of learning in a case-based reasoning system," M.S. program scholarly paper, Dept. of Computer Science, Clemson University, SC 29634, August 1994.
- b. Davis, J.S. and V. Jindal, "CBR system learning," working paper, Department of Management, Clemson University, SC 29634, July 1995.

3.0 *Test preparations.*

No security or privacy restrictions apply.

3.10 *Selection of test cases for the size prediction component.*

3.1.1 *Hardware preparation.* n/a.

3.1.2. *Software preparation.* As described in reference "a", determine how many test cases will be used, say 100. Using a database management system such as dBase IV, store a random number in each on-hand case. Sort the cases in order of that random number. Select the desired number (e.g. 100) from the first ones and store them in a separate file.

3.1.3 *Other pre-test preparations.* n/a

3.20 *Setting up new case-based reasoning library for size prediction component.*

3.2.1 *Hardware preparation.* n/a.

3.2.2. *Software preparation.*

As described in reference "a", import all but the selected test cases into a library using the ReMind case-based reasoning shell.

3.2.3 *Other pre-test preparations.* n/a.

3.30 *Running a test of the size prediction component.*

3.3.1 *Hardware preparation.* n/a.

3.3.2. *Software preparation.*

As described in reference "a", set up a batch file to automatically process each of the test cases and store results in a file.

4.0 *Test descriptions.*

4.1 *Test of the size prediction component.*

4.1.1 *Test case for the size prediction component.*

4.1.1.1 *Requirements addressed.*

A test case helps determine how well the system predicts clothing sizes.

4.1.1.2. *Prerequisite conditions.* n/a.

4.1.1.3. *Test inputs.*

Each test case contains the following data fields (measurement ranges are estimated based on data gathered at Fort Jackson):

1.	Height:	58 to	80
2.	Weight:	95 to	255
3.	Sleeve length:	25 to	40
4.	Waist:	23 to	48
5.	Seat:	30 to	50
6.	Breast:	30 to	50
7.	Head:	19 to	25
8.	Short sleeve shirt size		
9.	Long sleeve shirt neck		
10.	Long sleeve shirt sleeve		
11.	Black coat size		
12.	Black coat length		
13.	Green coat size		
14.	Green coat length		
15.	Trousers size		
16.	Trousers length		

4.1.1.4 *Expected test results.*

We expect the system to predict the same sizes contained in the test case.

4.1.1.5 *Criteria for evaluating results.*

One may evaluate the system based on its first, second or third choices for clothing sizes. When multiple test cases are processed, the system should achieve about the same percentage of correctness as human fitters do.

4.1.1.1.6 *Test procedure.*

Details are found in references a and b. A batch file should be created that causes multiple test cases to be processed, one by one, and the results stored in a file.

4.1.1.1.7 *Assumptions and constraints.* n/a

5.0 *Requirements traceability.*

Testing described herein is intended to determine how well the system satisfies the requirement to predict clothing sizes.

6.0 *Notes.* n/a

SOFTWARE USERS MANUAL—Size Prediction
(contract specifies DI-MCCR-8013; this outline is from DI-IPSC-81443,
published in MIL-STD-498, 5 December 1994)

1.0 *Scope.*

1.1 *Identification.*

This manual describes software completed in contract DLA900-87-D-0017, D.O. 0026, which extracts body measurements from a 3D body scan and predicts clothing sizes from body measurements.

1.2 *System overview.*

Clemson Apparel Research began the project to design an expert system for initial try-on of the US Army men's dress uniform began on June 10, 1992. This was a contract sponsored by the Defense Logistics Agency. The objective was to automate the prediction of US Army male dress uniform initial issue try-on size by employing an expert system in coordination with accurate 3-dimensional, non-contact body measurement. The history of system development is described in detail in the final report of which this users manual is a part. The Clothing Initial Issue Point (CIIP) at Fort Jackson, SC supported this project by serving as a test site.

1.3 *Document overview.*

The purpose of this document is to summarize how the software may be used. There are no security or privacy restrictions.

2.0 *Referenced documents.* (none).

3.0 *Software summary.*

3.1 *Software application.*

The software is intended to be used at a clothing issue point to speed up operations by automatically predicting clothing sizes. This project accomplished much of the work necessary to integrate a 3D body scanner and size prediction expert system into the uniform-issuing process of a CIIP such as the one at Fort Jackson. The missing link remains the 3D full-body scanner, which was not available during the project. The addition of a scanner would take care of the need for accurately taking a sufficient number of measurements for each soldier. The scanner would quickly capture a 3D body image. The project-produced software, which converts scanner output (a file of x-, y-, z-points) to specific body measurements would provide the data necessary (stored cases of measurements) for the size prediction software to determine the sizes to be tried on for issue. The current size prediction software works with manually-taken measurements as input.

3.2 *Software inventory.*

The system consists of two parts, measurement extraction and size prediction. The size prediction software requires the following files to operate, none of which have security or privacy considerations:

- Sizep.exe
(the executable main program)
- Sizep.ini
(includes user options)
- CBRemind.dll
(C libraries for the ReMind case-based reasoning shell)
- Jack.cbr
(case-based reasoning library; records of previously measured soldiers)

3.3 Software environment.

The system requires an IBM-compatible PC with 8MB Ram and 10 MB hard disk space free, and a printer, running Microsoft Windows 3.1 or later. CBRemind.dll (containing C libraries for the ReMind case-based reasoning shell) may be obtained from Cognitive Systems, Inc.

3.4 Software organization and overview of operation.

When the size prediction component is started, the user sees a blank form on the screen which accepts input describing measurements for a soldier. The computer accepts input as fast as it can be entered. It calculates measurements in approximately 40 seconds per soldier, with a 486 DX processor. The measurements can be printed by clicking on a "Print" button. The system's size predictions agree with those of human fitters 60-70% of the time.

3.5 Contingencies and alternate states and modes of operation.

If problems develop with the size prediction software, temporarily clothing sizes could be predicted manually.

3.6 Security and privacy.

There are no special security or privacy restrictions on the size prediction software, except the CBRemind.dll is copyrighted by Cognitive Systems, Inc.

3.7 Assistance and problem reporting.

Problems with the size prediction software should be referred to Professor Steve Davis at Clemson Apparel Research, 500 Lebanon Road, Clemson, SC 29670.

4.0 Access to the software.

The user should load CBRemind.dll in the C:\WINDOWS directory. Sizep.exe, Sizep.ini, and Jack.cbr could be loaded in the C:\SIZE directory. Then the user should edit the Sizep.ini file and modify entries as indicated by the directions in the file.

4.10 First-time user of the software.

4.1.1 Equipment familiarization.

The size prediction program uses the familiar windows interface and an ordinary personal computer. No special familiarization is necessary.

4.1.2 Access control. n/a.

4.1.3. Installation and setup.

The user should load CBRemind.dll in the C:\WINDOWS directory. Sizep.exe, Sizep.ini, and Jack.cbr could be loaded in the C:\SIZE directory. Then the user should edit the Sizep.ini file and modify entries as indicated by the directions in the file.

4.2 Initiating a session.

From the Windows main group, select File/Run, then enter "C:\SIZE\Sizep.exe C:\SIZE\Sizep.ini". If any error messages result, re-check installation of files as specified in 4.1.3. Also, examine the file Sizep.ini, which has self-contained instructions for setup.

4.3 Stopping and suspending work.

To exit the program, close the window that the program runs in. If there is an abnormal termination, the user will be notified with a message on the screen.

5.0 Processing reference guide.

5.1 Capabilities.

The size prediction software takes body measurements as input and provides sizes as output.

5.2 Conventions.

No special conventions are used.

5.3 Processing procedures.

5.3.1 Predict.

Clicking on this button signifies that all measurements have been entered and the user requests size predictions.

5.3.2 Print.

Clicking on this button causes the measurements and the size predictions to be printed.

5.3.3 Print and Clear.

Clicking on this button causes the measurements and the size predictions to be printed, and then the measurements are cleared.

5.4 Related processing.

If the user wishes to recalibrate the size prediction system, it requires building a new set of cases (a case consists of the body measurements of one soldier together with the garment sizes). These cases should be converted to ASCII format and then imported into the ReMind case-based reasoning shell. That shell then can be used to create a library. The new library would be used instead of the one called Jack.cbr, above. Details of this procedure are in the ReMind users manual.

5.5. Data backup.

There is no internal backup in the size prediction system. If one wishes a record of transactions, one could retain a copy of the printed output.

5.6 Recovery from errors, malfunctions, and emergencies.

In the event of errors or abnormal terminations, the user may re-boot the computer and then re-start the software.

5.7 Messages.

The only messages provided by the system during normal operation are progress messages such as "prediction trouser size...". A system message such as "General Application Failure" requires a recovery as mentioned in paragraph 5.6.

5.8 Quick-reference guide. n/a.

6.0 Notes.

CIIP —Clothing Initial Issue Point

ASCII —American Standard for Computer Information Interchange.

Appendix M

Measurement extraction CDRs

SOFTWARE DEVELOPMENT PLAN—3DM: Computer-Assisted Measurement Extraction

1.0 Introduction.

This document establishes the software development plan for 3DM, a computer-assisted measurement extraction system. This research and development project will develop new algorithms for determining body measurements from a 3D body scan.

2.0 Organization and Responsibility.

This project is conducted by Nancy Staples, Roy Pargas, and Steve Davis as part of the Clemson Apparel Research project. Nancy Staples is the principal investigator.

3.0 Management and Technical Controls.

Project members will meet at least once a month to discuss progress and problems. Monthly reports to the sponsor will be used partly as an internal management tool to document progress and identify any difficulties which arise.

4.0 Resources.

4.1 Personnel.

The following details the division of major responsibilities: Nancy Staples, project direction; development of schemes for measuring the body; Roy Pargas, development of algorithms and software for producing body measurements from the output of a 3D body scan (two part-time graduate students will assist Dr. Pargas); Steve Davis, development of an expert system which predicts garment sizes from body measurements extracted (two part-time graduate students will assist Dr. Davis).

4.2 Training. Davis and two graduate students should attend training in ReMind®, a case-based reasoning shell, no later than March 1993.

4.3 Data Processing Equipment. This project will employ IBM-compatible personal computers which are already on hand at the Clemson Apparel Research facility.

5.0 Software Development Schedule.

The project work is organized in two main efforts: 3D body measurement and expert system. The 3D body measurement tasks are: study of 3D scanners, development of basic measures, development of measurement macro language, and complete measurement software. The expert system tasks are: training in ReMind®, gather soldier data, develop expert system, and test expert system.

Gantt Chart

Item	Quarter	1	2	3	4	5	6	7	8
Management Plan									
Study of 3D Scanners									
Development of basic measures									
Development of measurement macro language									
Complete measurement software									
Training in ReMind®									
Gather soldier data									
Develop expert system									
Test expert system									

6.0 Risk Areas.

There are two principal risk areas. First, the project is not assured of getting a 3D body scanner in a timely fashion. This technology is still under development. Second, the project team is employing a new software technology for the expert system, case-based reasoning. None of the project team has any experience with this approach or with tools which support it. To minimize the risk concerning the 3D scanner, the project will: 1) search nationwide for potential developers of a scanner, and try to negotiate getting whichever product is first ready to be employed in this project, and 2) develop device-independent software for converting output of a 3D scanner into useful body measurements.

7.0 Monitoring and Reporting.

The primary scheme will be the monthly report to the project sponsor. This report will include status of program development, problems, risk areas, and planned solutions.

8.0 Documentation.

Since this project is exploratory, it will not attempt to develop documentation in a format appropriate for commercial software. Instead, the project will: 1) devote portions of monthly reports to documenting software, and 2) employ reports prepared by programmers as part of their requirements for an academic program.

9.0 Development Approach.

The purpose of this project is to develop a prototype system, so the software will be a prototype also. For the 3D measurement routines the project will use C++ and will develop a macro language to facilitate easy specification of various body measurements.

10.0 *Use of Existing Software.*

10.1 *Commercially Developed Software.* n/a

10.2 *Existing Applications Software.*

No software of this type will be used, with the exception of programs that may be furnished by the manufacturer of a 3D body scanner.

11.0 *Development and Test Tools.*

This project will develop a procedure for testing the expert system. The scheme calls for reserving a number of cases for testing, then comparing system predictions of garment sizes to the sizes associated with those cases.

12.0 *Security Controls and Requirements.*

Since results of this project are open to the public, no special security controls will be necessary.

SOFTWARE REQUIREMENTS SPECIFICATION—3DM: Computer-Assisted Measurement Extraction

1.0 *Introduction.*

This document establishes the requirements for 3DM, a computer-assisted measurement extraction system.

1.1 *Purpose.*

The intent is to prescribe clearly how the 3DM will perform.

1.2 *Scope.*

Requirements will be outlined.

1.3 *Terminology.*

3D Non-contact Measurement refers to a machine which scans the human body using light beams and outputs a file of x-, y-, and z-points from which body measurements can be extracted.

Case-based-reasoning (CBR) describes an approach to developing expert systems which employs previous cases to determine the correct disposition of a current case.

CIIP stands for Clothing Initial Issue Point.

1.4 *References.* (none)

1.5 *Overview.*

2.0 *General description.*

3DM will take as input a digitized image of the human body. The user will be able to take measurements interactively on this human image, including circumferential and distance measurements. In addition, the user will be able to write simple macros to automate the measurement extraction process.

2.1 *Product perspective.*

3DM runs on a SUN Workstation running the SUN OS/4 operating system. It is written in the language C++ and requires, as input, the digitized scan of a human body produced by a 3D scanner.

2.2 *Product functions.*

The system will interactively provide the user with measurements taken from a digitized image. These include circumferential measurements such as waist and chest measurements, distance measurements such as sleeve length, and radial measurements. In addition, a macro language facility allows the user to record the steps in taking a given measurement and play the recorded instructions back at a later time, thereby automating the measurement process.

2.3 *User characteristics.*

Intended users are personnel currently conducting measurement and clothing issue activities at military installations.

2.4 General constraints.

The system must be usable by people without specialized training.

2.5 Assumptions. (n/a)

3.0 Specific requirements.

3.1 Functional requirements.

INPUT: The only input required is a digitized image of a human body generated by a three-dimensional, non-contact scanner.

PROCESS: Using 3DM, the user is able to take measurements by pointing and clicking on different locations on the image and by making measurement selections from the menus provided.

OUTPUT: 3DM will generate the measurements selected, measurements such as waist and chest circumferences and sleeve length.

3.2 External interface requirements.

3.2.1 Human. None

3.2.2 Hardware. None

The system is a stand-alone system requiring a SUN Workstation running SUN OS/4 operating system.

3.2.3 Software. None

The system is a stand-alone system requiring a SUN Workstation running SUN OS/4 operating system.

3.2.4 Communications.

In a future version, it may be desirable to transmit soldier names and measurements electronically to clothing issue stations.

3.2.5 Location.

The workstation and printer will be located in the measurement room.

3.3.0 Performance requirements.

3.3.1 Input/output loads.

The system must handle 400 soldiers per day.

3.3.2 Database loads. n/a

3.3.3 *Response times.*

The system must produce measurements within 10 seconds after a measurement selection has been entered.

3.3.4 *Resource usage.*

Requires electrical power for computer system in measurement room.

3.40 *Design constraints.* None

3.4.1 *Standards compliance.*

Since this project will develop a research prototype, standards are not essential.

3.4.2 *Hardware limitations.*

System should run effectively on a SUN workstation running the SUN OS/4 operating system.

3.50 *Attributes (quality requirements).*

The system should take measurements as accurately as current human fitters.

3.5.1 *Availability.* (n/a; research prototype)

3.5.2 *Reliability.* (n/a; research prototype)

3.5.3 *Maintainability.* (n/a; research prototype)

SOFTWARE TEST DESCRIPTION—3DM: Computer-Assisted Measurement Extraction (contract specifies a Software Systems Development Test and Evaluation Plan, DI-MCCR-8039; this outline is from DI-IPSC-81439, published in MIL-STD-498, 5 December 1994)

1.0 *Scope.*

1.1 *Identification.*

This describes software completed in contract DLA900-87-D-0017, D.O. 0026, which extracts body measurements from a three-dimensional body scan and predicts clothing sizes from body measurements.

1.2 *System overview.*

Clemson Apparel Research began the project to design an expert system for initial try-on of the US Army men's dress uniform on June 10, 1992. This was a contract sponsored by the Defense Logistics Agency.

The objective was to automate the prediction of U.S. Army male dress uniform initial issue try-on size by employing an expert system in coordination with accurate 3-dimensional, non-contact body measurement. The history of system development is described in detail in the final report of which this is a part. The Clothing Initial Issue Point (CIIP) at Fort Jackson, SC supported this project by serving as a test site.

1.3 *Document overview.*

This describes how 3DM, the measurement extraction software system, may be tested.

2. *Referenced documents.*

Pargas, R.P. "Automatic Measurement Extraction from a 3D Scan of the Human Body", technical report, Department of Computer Science, Clemson University, SC 29634.

3.0 *Test preparations.*

No security or privacy restrictions apply.

3.10 *Development of list of body landmarks for measurement taking.*

A list of landmarks to be used to make measurements both for apparel sizing and made-to-measure will be prepared. A sample of these landmarks is listed below.

1. Neck center point
2. Neck left side
3. Neck center back
4. Neck right side
5. Right armscye bottom side
6. Right armscye top side (acromion)
7. Left armscye bottom side
8. Left armscye top side (acromion)
9. Shoulder center front

10. Chest center front (suprasternale)
11. Chest left side
12. Chest center back
13. Chest right side
14. Bust left bust point
15. Bust left side
16. Bust center back
17. Bust right side
18. Bust right bust point
19. Waist center front
20. Waist front left under shoulder
21. Waist left under bust
22. Waist left side
23. Waist back left under shoulder
24. Waist center back
25. Waist back right under shoulder
26. Waist right side
27. Waist right under bust
28. Waist front right under shoulder
29. Low hip center front
30. Low hip center back
31. Maximum abdomen center front
32. High hip center front
33. Right bicep center front
34. Left bicep center front
35. Right elbow outside (lateral epicondyle)
36. Left elbow outside (lateral epicondyle)
37. Right forearm center front
38. Left forearm center front
39. Right wrist outside (radiale)
40. Right wrist inside (ulnar styloid)
41. Left wrist outside (radiale)
42. Left wrist inside (ulnar styloid)
43. Right upper thigh inside
44. Left upper thigh inside
45. Right thigh center front
46. Left thigh center front
47. Right knee center front
48. Left knee center front
49. Right calf center front
50. Left calf center front
51. Right upper ankle center front
52. Left upper ankle center front
53. Right ankle inside (medial malleolus)
54. Left ankle inside (medial malleolus)
55. Crotch point
56. Right middle shoulder
57. Left middle shoulder

Once the landmarks have been defined, measurements will be defined based on the landmarks listed. Examples of possible measurements are:

Circumferences

1. Neck
2. Right armscye
3. Left armscye
4. Shoulder
5. Chest
6. Bust
7. Waist
8. Low-hip
9. Maximum abdomen
10. High-hip
11. Right bicep
12. Left bicep
13. Right elbow
14. Left elbow
15. Right forearm
16. Left forearm
17. Right wrist
18. Left wrist
19. Right upper-thigh
20. Left upper-thigh
21. Right thigh
22. Left thigh
23. Right knee
24. Left knee
25. Right calf
26. Left calf
27. Right upper ankle
28. Left upper ankle
29. Right ankle
30. Left ankle

Distances

31. Right over bust
32. Right bust to waist
33. Left over bust
34. Left bust to waist
35. Right shoulder to bust point
36. Left shoulder to bust point
37. Bust point to bust point
38. Front neck to waist
39. Right armscye to center-front waist
40. Left armscye to center-front waist
41. Right front shoulder to waist

42. Left front shoulder to waist
43. Right arm
44. Left arm
45. Right forearm
46. Left forearm
47. Right sideseam
48. Left sideseam
49. Right underarm
50. Left underarm
51. Right inseam
52. Left inseam
53. Front across shoulder
54. Chest width
55. Crotch depth
56. Back neck to waist
57. Shoulder blade width
58. Right over shoulder blade
59. Left over shoulder blade
60. Right armscye to center-back waist
61. Left armscye to center-back waist
62. Right back shoulder to waist
63. Left back shoulder to waist
64. Back across shoulder
65. Right shoulder width
66. Left shoulder width
67. Sleeve length
68. Stature

3.1.1 *Hardware preparation.* n/a

3.1.2 *Software preparation.* n/a

3.1.3 *Other pretest preparations.* n/a

4.0 *Test descriptions.*

4.1 *Test of 3DM.*

4.1.1. *Test cases for 3DM.*

4.1.1.1 *Requirements addressed.*

A test case helps evaluate how well 3DM takes measurements from a digitized form of the human body.

4.1.1.2 *Prerequisite conditions.*

A requirement of this test is that several samples of full scans of the human body have been successfully taken from a 3D scanner.

4.1.1.3. *Test inputs.*

Input to this test are the scans described in 4.1.1.2. Also required are the definitions of landmarks and measurements described in 3.10.

A comparison will be conducted between the measurements obtained using 3DM and measurements taken manually. The differences in measurements will be noted and analyzed and, if appropriate, modifications will be made to the computer software.

4.1.1.4 *Expected test results.*

3DM is expected to predict the same measurements obtained manually.

4.1.1.5 *Criteria for evaluating results.*

Statistical analyses will be used to measure how close the computer-assisted and manual measurements are to one another.

4.1.1.1.6 *Test procedure.*

The computer-assisted set of measurements will be compared with the manually-obtained set of measurements. Differences will be noted. The differences will be subjected to statistical testing, specifically whether the differences could possibly come from a normal population of values with mean zero. If so, then the conclusion will be that the differences are not significant and that the computer-assisted measurements will be judged equivalent to the manual measurements. If significant differences are found, a study of where the differences lie and modifications to the 3DM software will be made wherever necessary.

4.1.1.1.7 *Assumptions and constraints.* n/a

5.0 *Requirements traceability.*

Testing described herein is intended to determine how well 3DM satisfies the requirement to take measurements from a digitized human image.

6.0 *Notes.* n/a

SOFTWARE USERS MANUAL—3DM: Computer-Assisted Measurement
Extraction

See Appendix C