

AD-A138 115

CELESTIAL PATTERN RECOGNITION ALLOWING AUTONOMOUS
EARTH-SURFACE OR DEEP-S... (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... E P SCHEMPP
DEC 83 AFIT/GCS/EE/83D-19 F/G 1777

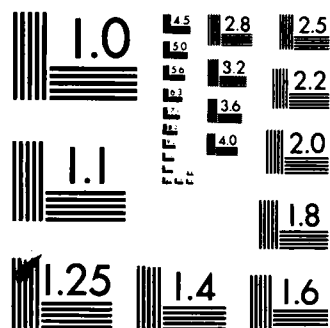
1/2

UNCLASSIFIED

NL

[]

[]



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA138115



CELESTIAL PATTERN RECOGNITION
ALLOWING AUTONOMOUS
EARTH-SURFACE OR DEEP-SPACE POSITIONING

THESIS

AFIT/GCS/EE/83D-19

Eugene P. Schempp
Capt USAF

DTIC FILE COPY

DTIC
ELECTE
FEB 22 1984
S E D

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

This document has been approved
for public release and sale; its
distribution is unlimited.

84 02 17 082

AFIT/GCS/EE/83D-19

CELESTIAL PATTERN RECOGNITION
ALLOWING AUTONOMOUS
EARTH-SURFACE OR DEEP-SPACE POSITIONING
THESIS

AFIT/GCS/EE/83D-19

Eugene P. Schempp
Capt USAF

1
DUG
S
E

Approved for public release; distribution unlimited

CELESTIAL PATTERN RECOGNITION
ALLOWING AUTONOMOUS
EARTH-SURFACE OR DEEP-SPACE
POSITIONING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Eugene P. Schempp, B.S., M.S.
Captain USAF
Graduate Computer Systems
December 1983

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



Preface and Acknowledgements

Today, there is no machine that can independently determine its location. This thesis explores the possibility of constructing such a device, and develops the logical processing necessary.

"No thesis is an island". Thanks go to Capt. James Garcia, who supported this project as its sponsor; to Capt. Lee Baker for lending file-transfer software and advice; to Mr. Dan Zambon for lending his operating system software, various equipment, and for his constant attitude of willingness to do anything to help the befuddled student; to Mr. John Miles for his photographic equipment and help; to Capt. Dean Pitts for his expertise with the oscilloscope and soldering iron; and to Dr. Matthew Kabrisky, whom I berated into being the advisor for this project, for his continual support (both moral and hardware), advise, insight, and his willingness to be a sounding board and constructive critic.

The great depth of my gratitude goes to Andrea Miles Schempp, my wife and my friend, who held us together when this project (and other AFIT enterprises) made me a husband and father in absentia. Throughout it all, she even managed to thwart my continual efforts to lose my sense of humor. This thesis is dedicated to her.

Contents

	Page
Preface and Acknowledgements	ii
List of Figures	v
List of Tables	vii
Abstract	viii
I. Introduction and Background	I-1
Background of the Project	-1
Background of the Concept	-3
Background on Celestial Positioning	-4
The Surface and Atmosphere of the Earth	-6
Positioning in Near Space	-9
Positioning in Deep Space	-12
Summary	-13
II. Preliminary Design Considerations	II-1
Geometry as a Feature Set	-1
Brightness as an Organizing Principle	-4
Data Base Considerations	-8
Field-of-View	-13
Sensor Configuration	-20
Resolution	-23
Refraction	-25
Planets	-25
III. Experimental Design	III-1
Data Base Construction	-2
Theory of Nested Recomposition	-7
Effects of Limited Accuracy	-13
IV. Program Construction	IV-1
Pattern Recognition Program	-2
Front End Processing	-12
Video Digitizer Driver	-15

Contents

V. Testing, Validation, and Results	V-1
Phase II Testing	-3
Phase III Testing	-13
Results	-15
VI. Conclusions and Recommendations	VI-1
Suggestions for Improved Algorithmic Performance	-1
Performance Summary	-2
Recommendations	-3
Bibliography	BIB-1
Appendix A -- Pattern Recognition Program Listing	A1-2
Sample Construction Program Listing	A2-1
Video Data Capture Program Listing	A3-1
Appendix B -- Data Base of Catalog Star Templates	B-1
Appendix C -- A Method to Determine Location in Deep Space	C-1
Appendix D -- A Program to Resolve an Earth-Surface Position without Azimuth Information	D-1
Appendix E -- A Brief Glossary of Terms Specific to This Project	E-1

List of Figures

<u>Figure</u>	<u>Page</u>
II-1 Construction of Mirror Image Polygons . . .	II-3
II-2a Chart of Fourth Magnitude Stars, Northern Hemisphere	II-14
II-2b Chart of Fourth Magnitude Stars, Southern Hemisphere	II-15
II-3 Image Zones in a Generic Sensor	II-17
II-4 Image Segmentation in a "Fisheye" Effect Sensor	II-19
II-5a Spiral Scan Example	II-21
II-5b Linear Array of Sensor Elements	II-21
II-6 Concept Diagram of a Passive Sensor	II-21
II-7a Sensor Detection of a Binary Star Gap . . .	II-24
II-7b Non-detection of the Gap due to Slight Sensor Repositioning	II-25
III-1 Sample Template Feature Vector	III-3
III-2 Recomposition of a Quadrilateral from Triangle Pairs	III-8
III-3 Recomposition Example	III-10
IV-1a Primary Flow in Pattern Recognizer Processing	IV-4
IV-1b Post-Triangle (Polygon-3) Processing . . .	IV-5
IV-1c Polygon-4 Level Pattern Recognition Processing	IV-6
IV-2a Output from 'Getpoints' Module	IV-13
IV-2b Output from 'Joinspots' Module	IV-14
IV-2c Output from 'Makesample' Module	IV-14

List of Figures

<u>Figure</u>	<u>Page</u>
V-1a A Sample from Testing Phase IIA	V-5
V-1b Primary Bodies Used in the Resulting Sample	V-5
V-1c Sample Data Submitted to the Main Program	V-6
V-2a A Sparse-Field Sample used in Phase IIA Testing	V-6
V-2b Primary Bodies Used from This Photograph	V-7
V-2c Resulting Sample Submitted to Main Program	V-7
V-3 "Improper" Selection of Candidate Star by Sample Construction Program . . .	V-14

List of Tables

<u>Table</u>	<u>Page</u>
II-1 Stars Quantized by Magnitude	II-13
V-1 Primary Program Decision Control Variables	V-4
V-2 Combined Results from Phase II Testing, Final Parameter Values	V-12

Abstract

This project explores the feasibility of using a small, general digital processor, with an optical sensor and a clock, as a means to autonomously determine location. A stellar pattern-recognition algorithm is presented that determines the identity of any of the navigation stars catalogued in the U.S. Naval Observatory Air Almanac which may be in the sensor's field of view. Appendices describe an iterative method for determining position on the surface of the earth from celestial observations without using azimuth information, and a mechanism for determining relative position in deep space.

I. INTRODUCTION AND BACKGROUND

Background of the Project

This research project is conceptual and opportunistic. It is conceptual in that it starts at the idea level, without any pre-existent theoretical or practical foundation or background to build upon. The idea is that an autonomous, quiescent, celestial positioning system can be constructed around an algorithm that uses a pattern-recognition methodology to identify specific celestial bodies. It is largely a feasibility study to determine if the logic can be automated which would allow an optical sensor and a small digital processor to derive its location.

Because the project starts as a concept, an effort was made to keep it as general as possible: to explore all reasonable avenues of development to avoid the premature rejection of any viable approach. This desire for generality results in a discussion of construction considerations for an actual device, and a treatment of the possible applicability of a pattern-recognition based celestial positioning device in different aspects of earth or space navigation, in addition to the development of the algorithm itself.

The opportunistic nature of the project comes from recent technological advances in photoelectronics, fiber-optics, and electronic microminiaturization. This last

factor, with its explosive decrease in the size, operating time, power requirements and cost of digital processors, is especially motivating for this project since an entire celestial-based positioning system could be constructed as a sensor plus a few chips. An opportunity is seen, therefore, to be able to construct a small, quiescent, completely autonomous positioning system, but only if the entire methodology of celestial positioning can be automated. Most of the celestial positioning method is well established: use of the stars for guidance dates to ancient mariners and explorers. But, as explained below, celestial positioning relies on knowledge of the identity of the celestial bodies used. This ability to identify the bodies before positioning measurements are taken from them has never before been present in a machine. The focus of this thesis, therefore, is to demonstrate an automatable method of identifying celestial bodies. This is accomplished by developing and testing a working algorithm, which is included in Appendix A. The creation of this algorithm forms the kernel of the project and of this report. The results of this celestial identification algorithm would then be used as input to either standard computations using spherical trigonometry (on the surface of the earth) or to other equations (for spatial positioning) to derive a position.

This report focuses on the segment of celestial positioning which is new: the celestial identification algorithm. As with many starting-level projects, the need

for context causes it to break ground over a fairly wide area, but it does not supply much depth outside of the main focus of the project. For example, sensor parameters are discussed, but only as a basis for evaluating the logical operations and organization of the pattern recognition algorithm, not to detail a specific sensor configuration for a particular mission environment. Other sections discuss alternative approaches and the usefulness of a device in different navigation regimes.

Both the research and this report have tried to follow these guidelines: first to produce a workable result, secondly to keep the approach and treatment general, and thirdly to keep everything as simple as possible.

I hope that the success that I have had with this project can lead to further development along these or parallel lines, with the result that an autonomous celestial positioning capability is considered a viable navigation system alternative for a multitude of missions.

Background of the Concept

Although this project does not build on other works, the basic idea of a star pattern-matching mechanism has been advanced before. Campbell, Dzilvelis, Packard, Potter, and Viglione (Ref 2; 7; 15; 17; 23) have published papers (most of these 20 years ago) suggesting that some form of star-identification algorithm, usually relying on the geometry of constellations or nearby stars, could be used for

5 navigational purposes. None of these, however, followed up their ideas with any published results showing that a solution could be obtained, with one exception: Packard (Ref 15) actually built a device which correctly found the Gemini twins. Curiously, Packard used an optical autocorrelation scheme, a method which is rotationally invariant. This makes autocorrelation a good technique for applications such as character recognition, where "C" and "U" (or "W"/"M", "d"/"p", etc.) have similar shapes, but because of different orientations of the shapes they represent different letters. Any method for identifying celestial bodies, however, needs to conclude that a pattern is the same despite any rotational variation. Unless a celestial positioning device is envisioned as operating in a dependent mode, i.e. it is first fed the orientation or direction of the body or the sensor's location (to then compute the orientation or direction), identification using autocorrelation seems inappropriate. The method developed for this project, nested recomposition, is examined in chapter III.

Background on Celestial Positioning

"For thousands of years man has relied on the stars as a primary source for determining his position on the surface of the earth." (Ref 21:1). The primary positioning methodology (on earth) has been to measure the azimuth and elevation of known celestial bodies, and then to use these

measurements to solve a problem in spherical trigonometry that yields a locus of points for each measured elevation. The locus is a circle centered at the sub-point of the body (see glossary) with a radius R that can be computed as

$$R = \text{archav} [\text{hav}(\text{long} - \text{LHA}) \cos(\text{lat}) \cos(\text{dec}) + \text{hav}(\text{lat} - \text{dec})]. \quad [1]$$

where

hav and archav	are the haversine and arc-haversine functions;
lat and long	are the latitude and longitude of the observer;
dec	is the declination of the body; and
LHA	is the local hour angle (celestial longitude) of the body.

Several such measurements, usually on different bodies, will yield several loci that intersect at the observer's position. But the identity of the body must be known to use this approach. Except in the special cases of the sun and moon (which are trivial to identify from earth) the normal technique is to start with a reasonably accurate initial position, which is used to calculate the relative altitude and direction of the desired body, and then to point the sensor in this direction and to assume that the nearest bright object is the one desired. If the assumption holds, then the elevation measurement will contribute to a refined position through the trigonometric calculations. But except when a human operator's star recognition ability is used, celestial positioning has always been performed in this manner: the approximate position must be known in order to

locate the desired body, in order to then refine or validate the position. This initial position usually includes some amount of error and, more importantly, requires that much of the navigation problem already be solved: that some reasonable estimate of position already be available before the celestial measurements are begun. This has prevented a celestial positioning methodology from ever being able to operate autonomously. But if a sensor could both determine the identity of a star as well as extract the normal measurement from it, it could independently determine position with only the correct time needed as input.

The usefulness of celestial navigation in general will vary with different missions, as will the importance of a star-identification subsystem. The next sections briefly discuss their advantages and limitations in three different navigation regimes:

- on or near the surface of the earth,
- in space in the vicinity of the earth, and
- in space beyond the inner solar system.

The Surface and Atmosphere of the Earth

Alternatives to Celestial Navigation. Although an ancient practice, celestial navigation still finds applications in such fields as shipping and military aircraft. The Air Force spends large sums to train officers in the techniques of manual celestial navigation, and these techniques form a regular method of navigating multi-engine

aircraft. Automated celestial positioning is sometimes available in the form of a "stellar inertial" system (such as the NAS-26 described in Ref 21). But because of the costs involved with such systems, their use is restricted to a handful of special purpose aircraft; and due to reliability considerations, two officers are normally employed, the first to operate the stellar-inertial system and the second navigator restricted to manual navigation techniques as a back-up in case of primary system failure.

Since other methods of navigation, independent of celestial techniques, exist on and near the earth's surface, the practicality of celestial based systems must be weighed against other navigation alternatives. Two types of systems that are excellent navigation methods (for many missions) are the inertially based navigation systems (INS) and the Global Positioning Satellite System.

The two primary attributes of celestial navigation are that it is passive (it has no electromagnetic or other emissions) and that its accuracy does not depend on the length of time it is used. This last feature differentiates it from an inertial navigation system, which is also passive. But the INS is a "dead-reckoning" device: it derives a present position by computing the distance and direction of travel, and then adding this distance vector to a prior position. It is incapable of correcting any error in the prior position, and since the distance vector cannot

be determined with absolute precision, an INS accumulates error over time. But the inertial technology has been operational for over thirty years (Ref 21) and is sufficiently accurate for many missions of limited duration (such as ICBM flights) and inexpensive enough to be employed on small vehicles (such as air-launched cruise missiles: Ref 19).

In contrast to the technical maturity of an INS, the Global Positioning Satellite System (GPSS) is not yet operational. But unlike an INS it is a "fixing" system: it determines position independent of any prior position, and therefore the accuracy does not degrade over time. The GPSS also will be a relatively inexpensive system to use; although the US government will spend billions of dollars to place the necessary 24 satellites in specific geosynchronous orbits, a signal receiver/location device should not be expensive, and the accuracy is expected to be extremely good, on the order of 10 meters in three dimensions (Ref 19). But military enthusiasm for heavy dependence upon GPSS is limited because, besides being not yet operational, the system may not be survivable in wartime.

Like an INS, celestial navigation has unquestioned wartime survivability. Like the GPSS, it is a "fixing" device that does not degrade over time. But the normal accuracy of celestial positioning is not as good as that of an INS for short periods of time (less than about five hours) and will probably not be nearly as good as the

eventual GPSS network.

Celestial observations can also be hampered by cloud cover. A sophisticated sensor that might be able to compensate for some amount of cloud cover attenuation would probably be too expensive to use in a simple positioning system. And although aircraft are subject to overhead cloud cover much less frequently than ground observers, clouds can still occasionally preclude the function of a simple sensor at normal flight levels.

Autonomous Celestial. Besides these limitations on celestial positioning in general, a star-identification device would have the additional disadvantage of the need to contend with sunlight half of the time. Astrotrackers, such as the Northrop NAS-26 military system, compensate for sunlight but use a narrow field-of-view (40 seconds of arc: Ref 21:5). A scanning sensor with a similar field-of-view could be used for the star-identification algorithm as well, but the much larger field-of-view that would be advantageous for a night-sky sensor (see chapter II) would have greater problems with sunlight.

Positioning in Near Space

General Celestial. This background light problem disappears in space, where the stars are continually visible. And the problem of cloud cover similarly does not exist, making celestial navigation a continuously available option to supplement or replace present procedures.

Spacecraft currently rely on positioning data from earth-surface stations for most of their mission durations. This has resulted in a heavy burden on the earth stations that has motivated numerous efforts into an autonomous satellite navigation capability (Ref 5; 11; 13; 17; 20). Celestial positioning has been considered as one way of alleviating this dependency, and Martin-Marietta Corporation has developed and tested a "Space Sextant" advanced as a practical solution (Ref 5:224). But celestial observations must have greater precision in space to afford a level of accuracy similar to the earth observations. This requirement stems from the fact that the error incurred is proportional to the distance of the observer from center of the earth (for the normal trigonometric solution). On earth this is about 3444 nautical miles, but a geosynchronous satellite orbits at 6.6 times this distance, and any error is multiplied by a similar factor. Consequently, the Space Sextant is a large device (weighing upwards of 25 kg and requiring around 50 watts) capable of precise (1 second of arc) measurements (Ref 5:224). This level of precision, used with the standard geocentric equations, gives a positional accuracy of 100 to 1000 feet (the exact value varies inversely with the spacecraft altitude). This accuracy is still inferior to GPSS, which should work as well in near-space as on the earth's surface.

Autonomous Celestial. An autonomous capability, i.e. a system that could independently acquire, identify, and

measure celestial-body parameters, would be required for celestial positioning to be a practical alternative for reducing the amount of ground control needed by vehicles in near-space. A positioning system that required an external sensor-pointing signal would offer no advantage over the present system of earth-based positioning signals. But the practical value of such a system would depend somewhat on the characteristics of the orbit. A high-earth orbit would suffer the greatest amount of accuracy degradation from the geocentric solution (such as equation 1). If the resulting precision is acceptable, however, the higher orbits would have the least need for a stellar pattern-recognition program since the easily identified sun, moon, and earth would all be visible about 90 per cent of the time. (At a geosynchronous altitude of 19,300 nautical miles the earth subtends 17 degrees of arc, so it will not usually block the sun or moon from view). If the altitude is known then, similar to the procedure on the earth's surface, "elevation" measurements of the sun and moon (using the earth's center, or the illuminated limb with a correction, as the measurement reference) will give two circles-of-position that will normally intersect at two "points" (within the tolerance of the measurements). This remaining ambiguity can generally be removed, since sufficient positional information now exists to point the sensor in the direction of a known star or planet. A bright body (outside the plane

of the ecliptic so that it is fairly isolated) should be able to be located without the need for a pattern-matching algorithm. This body can be used for a third circle-of-position to completely determine the location. Since a non-ambiguous position estimate now exists, it can be used (if desired) with the ephemeris to point the sensor at another body (whose relative direction can now similarly be computed without the stellar pattern-recognition), and so on. However, both celestial positioning and independent star identification would be more valuable in a low earth orbit. Due to the smaller distance from the earth's center (compared to the high orbit) the obtainable precision is much better: it is about 87 to 96 percent of earth-surface precision for altitudes of 100 to 500 nautical miles. And because the earth now obscures a large fraction of the spacecraft's "sky" (it can subtend up to 153 degrees) either the sun or the moon (or both) would be blocked most of the time, preventing their joint use to provide the initial steps for position resolution. A method to identify additional bodies, i.e. the true stars, therefore becomes most important in the lowest orbits, where celestial positioning provides the greatest accuracy.

Positioning in Deep Space.

General Celestial. In this discussion, a spacecraft will be said to be in deep space (as differentiated from near space) when it no longer can simply identify the sun as

significantly different than other stars. Today the first man-made probes are entering this regime of space. At this distance the GPSS system becomes only a point source of energy, and unable to provide a position. Inertial systems cease to function in free-fall flight, and would also be impractical due to their constant requirement for power and the unbounded errors that accumulate on long duration missions. Since there is little else in deep space to use as a reference, navigation would probably need to be based on celestial observations. The use of pulsar timing, an interplanetary beacon, very long baseline or connected element interferometry, and an orbiting deep-space radio network have been discussed as potential navigation methods (Ref 1:397). But these methods are all lacking autonomous or operationally verified capabilities. If any of these are proven suitable for some deep-space missions, a celestial system may still be required as a supplemental method, or the radiometric method could supplement the celestial.

At this distance from the earth, no reasonable accuracy is obtainable from the spherical trigonometric calculations as performed on the earth's surface. Alternate positioning computations, for example those discussed in Appendix C, would also provide degraded absolute accuracy, but the relative accuracy should be sufficient for purposes of interstellar navigation.

Autonomous Celestial. The need to make the navigation capabilities self-sufficient is as essential in deep-space as the need to use celestial. A star-identification technique, through pattern recognition or some other methodology, would seem the only alternative to the problems of transmitting and receiving earth-based sensor-pointing signals over tremendous distances to unmanned vehicles. For manned deep-space missions, autonomous star-identification might be less necessary, but would still seem highly desirable.

Summary

With the addition of a working star-identification algorithm to enable autonomous operation, celestial navigation may be able to augment or even supplant, for some applications, other navigation systems. On the surface of the earth its applicability for any mission would depend on practical considerations such as cost, size, power, speed, and reliability because of the competition from other navigation media. For many missions, limitations due to sunlight or cloud cover could restrict it to a supplemental or backup role. These limitations would not be present in near-space, and autonomous celestial positioning using an automated star-identification algorithm may be an option for low-earth orbit vehicles. For spacecraft or satellites in high altitude earth orbits, however, a star-pattern recognition algorithm might be unnecessary by recognizing

and utilizing the sun, earth, and moon. The probable value of a recognition algorithm (and celestial navigation generally) increases, however, as mankind and his sensors extend outward from the earth, and may be essential in confronting the vast mission durations and emptiness of deep (interstellar) space.

II. PRELIMINARY DESIGN CONSIDERATIONS

It is not the purpose of this project to suggest a particular design for a celestial navigation system. Although the digital processing facets of different celestial systems can be the same, different missions in different regimes with different operational requirements will probably necessitate different hardware configurations for the sensor. However some thought regarding the uses and limitations of a generic sensor can serve as guidelines for the software design of the processor.

Geometry as a Feature Set

Humans have long been able to recognize individual stars. Aided by imagination, they grouped the celestial sphere into constellations and embellished the patterns with stories to aid their memory. The classic mythological or astronomical constellations, however, are difficult for a machine system to use because of their non-uniform size and arbitrary boundaries. The approach followed in this project uses the geometry provided by the patterns of points of light in the sky, but operates within the constraint of a fixed, limited field-of-view ("FOV") since the simplest optical sensors can then be employed. The N stars in the FOV are treated as the vertices of a two-dimensional convex polygon having $N(N-1)/2$ adjoining sides. Either the

relative lengths or the absolute lengths of these sides can form a possible feature set. The absolute lengths are used herein because they provide greater discrimination per number of vertices, and would be available from either a fixed focal-length sensor (one without a "zoom" capability) or a variable focal-length sensor with the proper focal length scale factor applied to any measurements.

The term "polygon" in this discussion refers to any two-dimensional figure composed from N vertices. This differs from the normal geometric definition that counts sides, since a quadrilateral will have four vertices and a total of six "sides" counting the interior sides. It will still look, however, like a normal quadrilateral. Also, when a number is appended to "polygon" it denotes the number of vertices, so a "polygon-4" is a quadrilateral.

For a star-identification problem, any feature set needs to be rotationally invariant. Therefore the actual positions of the vertices on any reference frame, as well as all angular measurements, are ignored. Using only the data on the lengths of the sides is insufficient to uniquely determine a polygon, but the side data alone does establish membership in a set of only two possible solutions, with the solutions being mirror images of each other. Figure II-1 shows an example for the simplest case, a triangle. Construction of higher-order polygons is analogous: there are exactly two ways that a third distance value can join

the first two values, and once one of these is selected the three vertices uniquely specify the position of all subsequent vertices.

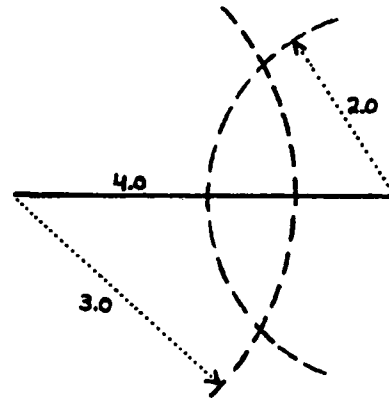


Figure II-1a: Given three side lengths (e.g. 2.0, 3.0, 4.0) any one side may be plotted first. Then each of the remaining lengths describes an arc from one end of the plotted side.

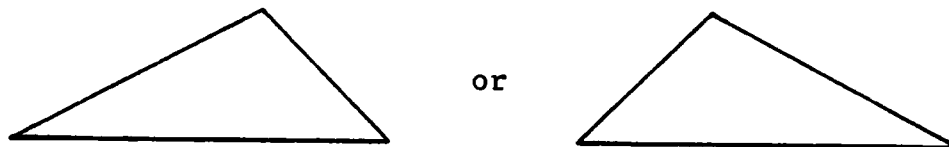


Figure II-1b: The two possible results are determined by the two points of intersection of the arcs.

Other methods, using angular measurements, can also define the polygons (such as a triangle by two sides and the included angle). But limiting the feature vector to the single component type "side length" speeds and simplifies the logical construction and processing of the algorithm.

Using side-length data alone does allow the possibility

of the incorrect "match" of a pattern with its mirror-image pattern. The frequency of this faulty conclusion decreases with higher polygons, however, and it proved non-fatal in the testing phases.

Planar geometry is not the only possible feature set. A correlation of the absolute or relative visual magnitudes of stars in the FOV is another option, discussed later. Or an array of sensors centered at different frequencies, to take advantage of the spectral diversity of the different types of stars, might provide a feature set that allows sufficient differentiation of a limited number of individual stars if the number of discrete sensors is large enough. This approach was not used in this project because of the cost of a multi-sensor array compared to a single sensor and microprocessor device, but is discussed in Reference 8.

Brightness as an Organizing Principle

Celestial positioning needs only a limited number of measurements taken from known bodies to resolve a position. On the surface of the earth a standard technique involves measurements taken from three bodies used together. The three resulting circles of position will intersect at a single point (subject to accuracy limitations) thereby avoiding any positional ambiguity, and also provide a mutual cross-check of the observations, and (if chosen carefully) will further provide a method of cancelling out any elevation measurement bias. Both the observed elevation and

the true (or relative) azimuth are typically used for position determination, but as shown in Appendix D the elevation data alone can uniquely determine an earth-surface location. (More than three measured bodies can be used, but generally serve only to further validate a position).

Given, then, that only a small number of celestial bodies need to be used per location, an obvious criteria for choosing between different bodies in the field-of-view would be the brightness of the objects. The brighter bodies are generally the easiest for any sensor to acquire and measure, and under the variable atmospheric conditions on the earth they are the ones that will be optically detectable the largest percentage of the time.

Brightness, therefore, is used as a main principle of organization in this project. It is the brightest body in an arbitrary FOV (or segment of a FOV) that the algorithm first tries to identify, and the other stars in that FOV (or segment) are processed in the order of their relative brightness. This places a requirement on the sensor to be able to adequately differentiate the brightness levels of the bodies. While this should be a very realizable sensor specification (the general-purpose television camera used for overall system validation was adequate in this regard) most of the problems that arose in the development phases of the project were special cases of an inability to perfectly rank-order stars of nearly equal visual magnitude (chapter

IV). But some level of brightness differentiation seems necessary. The heavens comprise a very non-uniform distribution of stars of each magnitude, and a sensor must be sensitive enough to acquire a sufficient number of stars in the sparser regions of the celestial sphere to generate a usable feature vector. A sensor that is adequate for the sparse regions, i.e. can sufficiently cover "sparse-field" template stars, will find too many stars in its FOV to be reasonably processed when pointed at a dense region of the sky. Other proposed schemes (Ref 2; 7; 17) have suggested a gain-control feedback loop from the processor to adjust sensor sensitivity until a "proper" number of stars are in the sensor FOV. This incurs a penalty in processing overhead and time: the entire FOV must be processed to determine the number of bodies therein, and in most cases multiple iterations of change-the-sensitivity/count-the-points must be performed. And the basic requirement still exists for a feedback-type system: if an algorithm requires N stars to be present in the field-of-view, the sensor must be able to differentiate between the N th and the $(N+1)$ st star. When these two stars are of nearly identical luminosity, or when one or both are periodic or non-periodic variables, or if the sensor response or atmospheric attenuation is not exactly uniform, the N th star may be excluded and the "incorrect" star $(N+1)$ included. Any algorithm must be able to accommodate this type of occurrence.

The method used for this project is to assume that a sensor's sensitivity level is fixed at (at least) the level sufficient for adequate generation of a sparse-field star's feature vector. Then all of the bodies that are found in any FOV are accepted (i.e. none are "tuned out"). But for a heavily populated FOV only the brightest M are extracted, in order of their brightness, to be processed as vertices. (M is a modifiable program parameter that was never set greater than 9). Since the capability to differentiate the Nth and (N+1)st brightness values is needed by any method at least once, this algorithm uses the capability to rank-order all of the brightest M bodies in the FOV. Incorrect orderings (those that differ from the sequence in the database) were a frequent occurrence: they were found, to varying degrees, in every sample used in phase II and phase III testing (chapter V). However the algorithm was made flexible enough to handle these misorderings.

To minimize the impact that brightness differentiation had on algorithmic performance, the identification program originally used only a geometric feature set as the vector for individual stars. However, since the relative brightness values were in most cases easily measured, and since this was an additional information source that could be used for template discrimination, a brightness correlation procedure was added in a later stage of algorithm development (chapter V). However, the geometric

relationships always operated as the dominant pattern-recognition feature.

Data Base Considerations

A preliminary question is "what set of stars should be catalogued as templates, that is, what is the list of stars that we should try to match to input samples?" The size of the template-set of catalog stars is a compromise between an algorithm's speed and the device's chance of a successful identification. A small number of templates in the database might appear to require a smaller feature vector as well, since less power of discrimination is required to separate fewer articles. But this conclusion is false: a star catalogued as a template in the database must be differentiated from all other celestial bodies in the heavens, not just the other catalogued stars, so the discriminatory power of the feature vector cannot shrink with the database.

The gains in ease and speed with a small database will be offset by an increase in the probability that the sensor FOV will not contain one of the catalogued stars. This probability is a direct result of the size of the FOV (discussed in the next section), but for all except very large FOV sensors, the number of catalogued stars in the database is proportional to the frequency of having a catalogued star included in the sensor's field-of-view. To

some extent, the reduced processing time from a reduced database can be used to offset the lower inclusion frequency, since after a recognition failure the sensor can be pointed at a second area of the sky and begin processing again, while a larger database system with the same FOV would still be processing the first image (but with a greater probability of finding a catalogued star therein). Actual mission requirements would be a major consideration in the number of stars stored as the database templates.

Because this project focused more on showing that a solution was possible than on performance optimization, a pre-existent set of 57 stars, numbered and catalogued by the U. S. Naval Observatory in The Air Almanac (Ref 14) was used as the template set, rather than a set tailored to the performance of the algorithm. The two salient characteristics that led to inclusion in the Almanac set are coverage of the celestial sphere and brightness. Because the stars are used for navigation, they were selected from each portion of the sky so that several would be visible from any point on earth at all times. All of the first magnitude stars except Crux beta (whose proximity to Crux alpha and gamma make it redundant) are included, with the dimmest included star (Eridanus theta) having a 3.1 magnitude.

Along with the inclusion question, it is necessary to select the parameters used to create a feature set for each

of the catalogued stars. These include the number, arrangement, brightness, and proximity of the other "support" stars.

Using a geometric feature set, each of the catalog-stars is one vertex of an $(N+1)$ -polygon, where N is the number of other stars in the region of the catalog-star that comprise the other vertices. A constant value for N can be selected to apply to all of the templates, but this approach tends to penalize either rich-field or sparse-field stars (see Glossary: Appendix E). If N is a low number (such as three or four) many of the easily detectable and measurable support-stars surrounding a rich-field catalog-star will be ignored, wasting usable discriminatory geometric pattern information, and in effect pulling down all templates to the level of the catalog-star with the sparsest support field. Conversely, a large number for N will require a sensor to be able to detect and brightness-differentiate very faint stars (to at least the sixth magnitude for some of the sparse-field templates used). To avoid choosing one of these alternatives, N is treated only as an upper bound on the number of support-stars per template. The actual number used is a function of the brightness of the other stars in the region, and varied in the database from a low of three in a very sparse field to a high of nine. The number nine is essentially arbitrary, motivated by the desire to include excess information in the feature vector rather than too

little (in the rich-field cases), and the feeling that optimization of processing speed could come at a later time: the first priority was a workable algorithm, and success was more probable with an over-abundance of data. Nine support-stars seems more than sufficient as an upper bound: a convex polygon of $N+1$ vertices (the catalog star plus N support stars) will have $N(N+1)/2$ interconnecting sides. When N is a maximum (nine), the resulting feature vector contains 45 distance measurements. This provides plenty of discrimination between the 57 templates, and also proved sufficient to eliminate non-catalogued stars.

Allowing the number of support stars to remain a variable is motivated by the natural phenomenon of non-uniform star distribution in the celestial sphere, but it caused a very serious conceptual problem. If N is a variable (three to nine) then the number of values in the template feature vector is a variable (from six to 45), and no theory or application of pattern-recognition methodology could be found where template size was anything other than a pre-specified constant. The approaches in the literature generally apply some distance rule (Menkowski-one or two, Itakura, etc.) to a sample vector, on an element-by-element basis, with the corresponding elements of the template vector. But variable size template vectors (further confounded by sample vectors also of variable dimension: see chapter III) destroys the notion of "correspondence". The

concept developed to handle this problem, called "nested reconstruction", is presented in chapter III. This allowed the selection of only "good" support-stars for each particular template.

A "good" support-star is near the catalog-star and reasonably bright. A preliminary criteria for "nearness" is obviously that it is in the same effective field-of-view. Since more bright stars are available as the size of the FOV increases, the support-star "inclusion radius" should be as wide as possible. However, it is limited to less than one half of the FOV for reasons covered in the next section. In this project the support-star inclusion radius (or "support-radius") used to construct the database was set at less than or equal to ten degrees of arc from each particular catalog-star.

Placing a minimum limit on the brightness level of the support-stars allows the sensitivity of the sensor to be specified. A higher minimum level allows a simpler sensor, and since a large field-of-view and good resolution are both desirable sensor characteristics but generally incommensurate with light-gathering ability, the highest usable minimum level is desirable. Since the human eye can readily detect fourth magnitude bodies, and there are 914 stars of the fourth magnitude or brighter (Table II-1), the original cut-off level used was 4.5. This level was dropped slightly, to a 4.6 level (and in a single instance to 4.8) to accomodate some of the sparse-field catalog-stars.

Table II-1: Stars Quantized by Magnitude.
(modified from Ref 7)

Magnitude	! Number of This ! Magnitude	! Inclusive ! Total	Stars per ! Square Degree
1	21	21	0.000509
2	65	86	0.00208
3	202	288	0.00698
4	626	914	0.0222
5	1939	2853	0.0692
6	6012	8865	0.215

Field-of-View

Determination of the sensor field-of-view is the system parameter that probably has the greatest overall effect on system behavior, as well as being the one most sensitive to performance tradeoffs for various missions. It is also closely related to the considerations on database construction just discussed.

The ideal optical sensor would have good low-light-level sensitivity, high resolution, the ability to filter out background light if present, a high probability of containing an identifiable body, and a sufficient number of other stars to provide a good supporting-geometry feature vector. The first three of these attributes are most easily realizable in a small field-of-view device, while the last two specifications each argue independently for a field-of-view as large as possible.

The Northern Sky

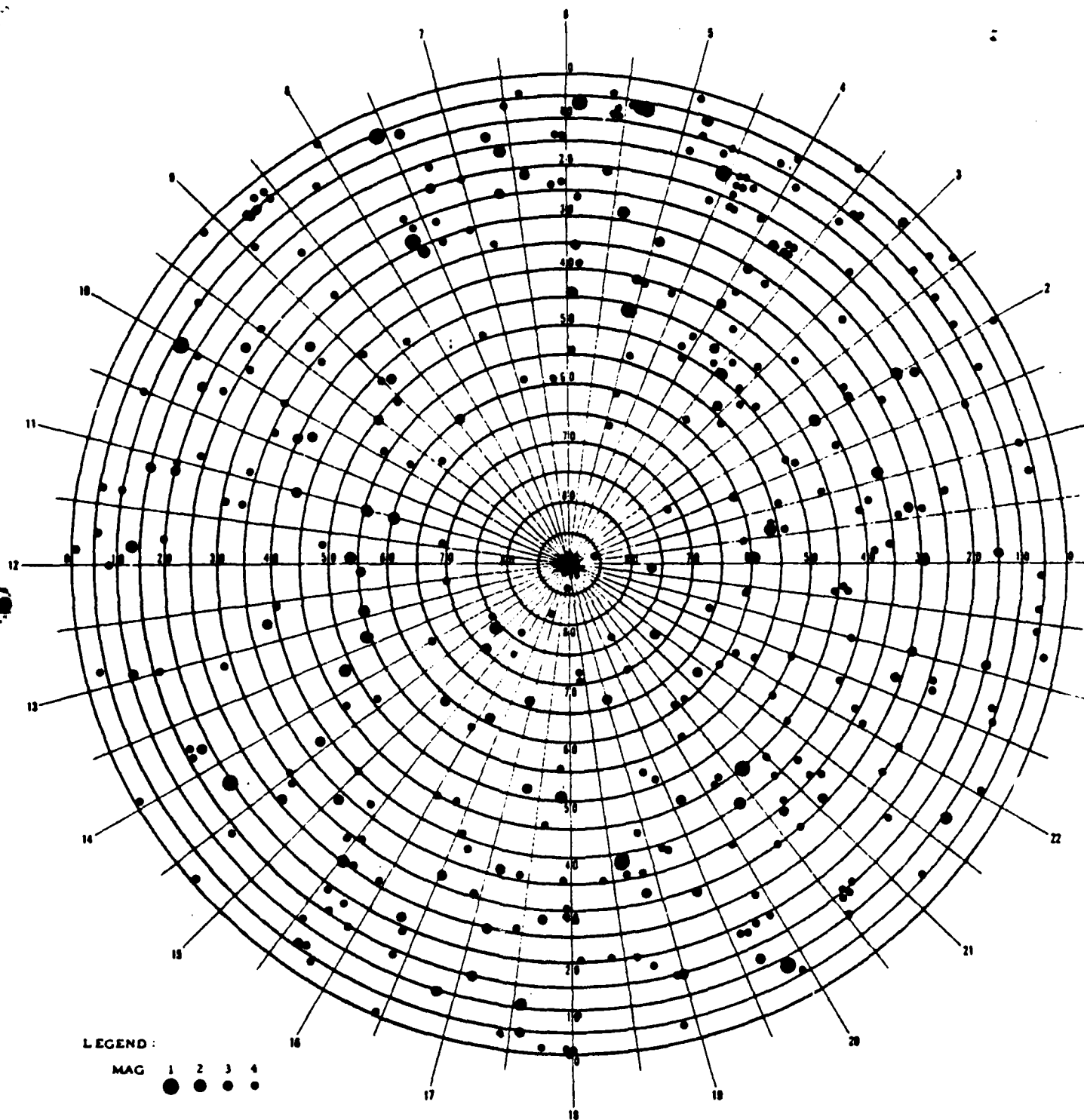


Fig.II-2a. Star Chart for Stars of 4.5 Magnitude or Brighter

(from Ref 9)

The Southern Sky

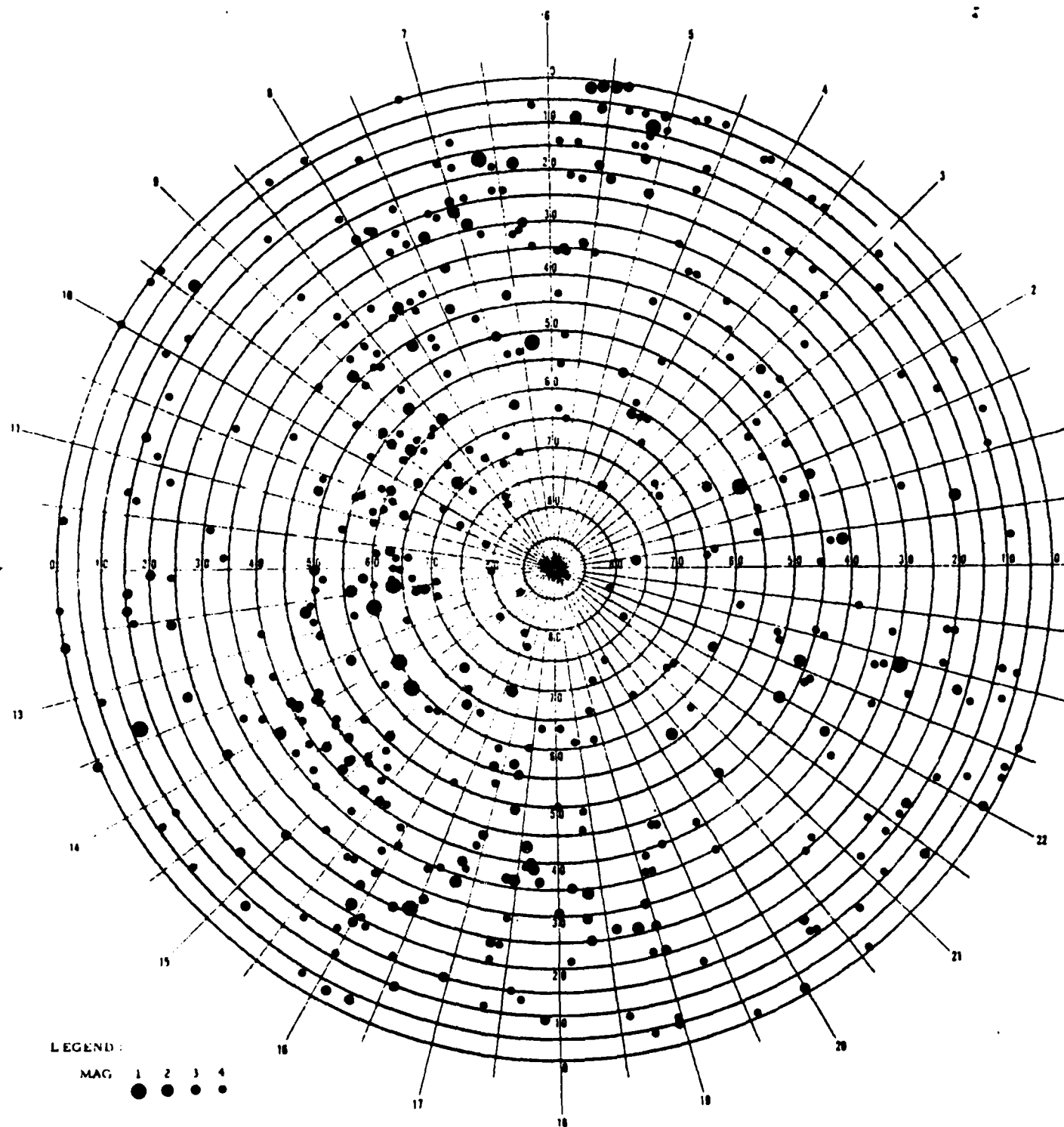


Fig. II-2b. Star Chart for Stars of 4.5 Magnitude or Brighter
(from Ref 9)

Despite the attempt by the Naval Observatory to select cataloged stars from all segments of the celestial sphere, some sectors had few if any bright stars. For a sensor to have a theoretical 100 per cent probability of containing a catalog-star in its field-of-view, it would need to span at least a 42 degree arc, since the greater Camelopardus area of the northern polar region (bounded roughly by Ursa Major, Ursa Minor, Perseus, Cassiopeia and Auriga) has an area with a 21 degrees radius that does not contain a catalog-star. Expanding the data base to include as templates all of the stars through the fourth magnitude would still leave a gap of approximately 15 degrees in Camelopardus, necessitating a sensor FOV with a minimum diameter of 30 degrees of arc for a 100 percent assurance of catalog-star capture. Of course, a capture rate of less than 100 per cent is probably acceptable for practically any mission: if any FOV turns up empty, point the sensor at another region. But as the FOV shrinks the number of empty regions grows. And the field-of-view numbers discussed so far are only the inner-zone values: the total FOV must include both this distance and an outer-zone figure to accomodate the region around each catalog-star that includes the support-stars. The outer-zone is the parameter mentioned in the previous section as the inclusion radius, set to ten degrees for the database constructed for this project. But to include all of the support-stars for a given catalog-star, any FOV must be at

least twice this outer-zone value (see figure II-3). In the case where the outer-zone value is ten degrees, as with this database, the minimum FOV that could be sure of including all of the support-stars would be 20 degrees.

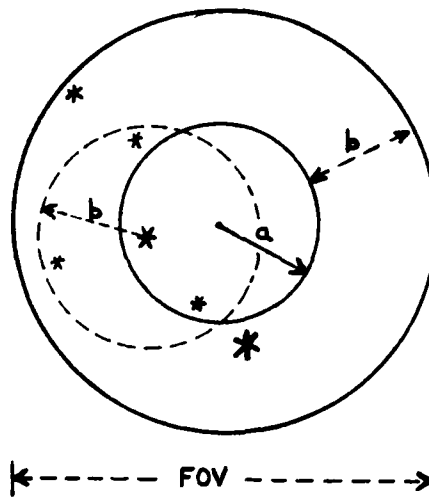


Figure II-3: Image Zones in a Generic Sensor.
 "a" is the inner-zone radius,
 "b" is the support-radius,
 $FOV \text{ (field-of-view)} = 2(a+b)$.

But this case would imply an inner-zone radius of zero; i.e. using a 20 degree total FOV would succeed only if the catalog star was at the exact center of the image. A feedback loop to the sensor to place the brightest body into the exact center would generally solve this requirement, but a simpler sensor with a larger FOV could simply pick the brightest object in the inner-zone as the candidate star, avoiding the time and hardware needed to center the body.

The conclusion from the inner-zone and outer-zone requirements is that the sensor field-of-view should be as large as is practical, up to as much as 150 degrees.

Smaller FOVs are certainly usable: the primary validation phase for this project used images that spanned approximately 24 degrees of arc. But the smaller FOV reduces the probability of a catalog-star in the inner-zone, or at least forces a trade-off between this factor and a reduced pattern-discrimination ability from a smaller outer-zone (support-radius) value. All of this is, obviously, highly dependent on the number and distribution of the catalog-stars stored as templates in the database.

There is no theoretical need for an upper bound on sensor field-of-view (except for horizon and refraction considerations on the earth's surface). If a very large FOV is obtainable, such as the entire sky within 60 degrees of the sensor's zenith (making the total FOV equal to 120 degrees: a "fisheye lens" effect) three immediate advantages accrue. First the FOV can be segmented by software into a number of regions equal to the number of bodies required for position determination (for example, three) with no need to reposition/repoint the sensor (figure II-4). Secondly, the wide inner-zone diameter (30 degrees) could guarantee the presence of a catalog-star with even a small number of templates in the database. The 57 catalog-stars from the Air Almanac would not guarantee template inclusion because of the Camelopardus gap, but in this case resegmenting the same image (rotating the segments 45 degrees in either direction) would then guarantee inclusion.

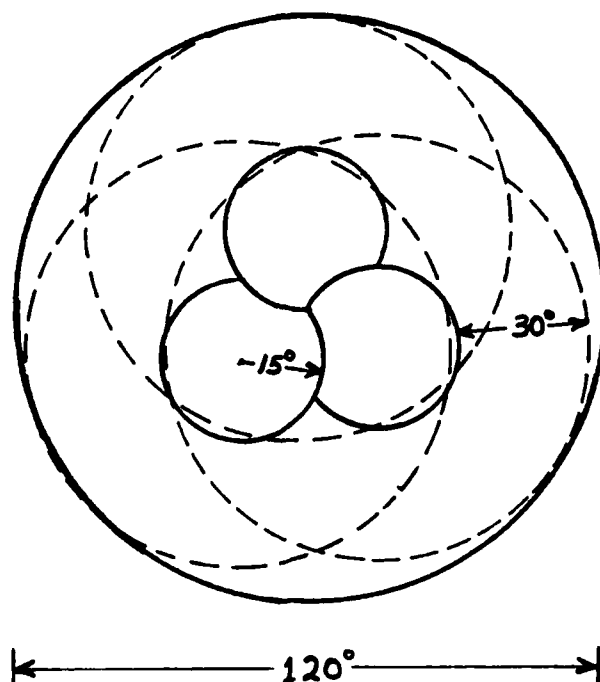


Fig. II-4: Image Segmentation with a "Fisheye" Effect System.

With this inner-zone diameter, and rotating segmentation as needed, a smaller database (comprised of perhaps only the first magnitude stars) would be a possibility. Last among the advantages, the large support-star inclusion radius (30 degrees) effectively eliminates any stars from the sparse-field classification as it applied in this model. While the number and quality of support-stars will still vary, recruiting from a region nine times as large assures in all cases a good supply of low magnitude support-stars.

A final observation regarding the sensor field-of-view is that the foregoing discussion deals with what is actually a resultant FOV that may be composed of multiple responses from a small FOV sensor. Re-using a limited FOV device could help to resolve the specification clash that called for high resolution and sensitivity but also a large

effective field-of-view. However, some difficulty is preserved in this method, related to scan time and sensor configuration.

Sensor Configuration

Even more than other design considerations, the method of sensor operation must be sensitive to operational requirements such as speed of operation, presence of background light, platform movement, and so on. The various possible configurations can be grouped into two categories: active or passive. Most of the papers that addressed this issue (Ref 2; 7; 17) envisioned an active sensor, typically a single vidicon/orthicon element that traversed a set field, such as the raster scan of a television. Campbell (Ref 2) suggested a spiral scan that progresses outward from a candidate star, thereby converting the angular and distance relationship of the surrounding bodies to a function of scan time (figure II-5a). M. Kabrisky suggested a modification to this: that using several elements in a linear arrangement would cut scan time to a fraction and also allow some parallel processing (figure II-5b). All active sensors, however, take some time to compose an entire image. If the device or vehicle moves during the image accretion time interval, errors will be introduced into the resulting image. The speed of a normal TV raster scan is probably sufficient for normal movements in most applications, but a scenario such as a rapidly tumbling

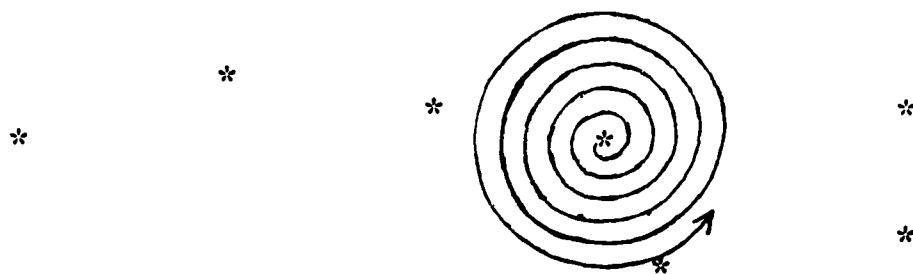


Figure II-5a. Spiral Scan Example. (from Ref 2).



Figure II-5b. Linear Array of Sensor Elements.

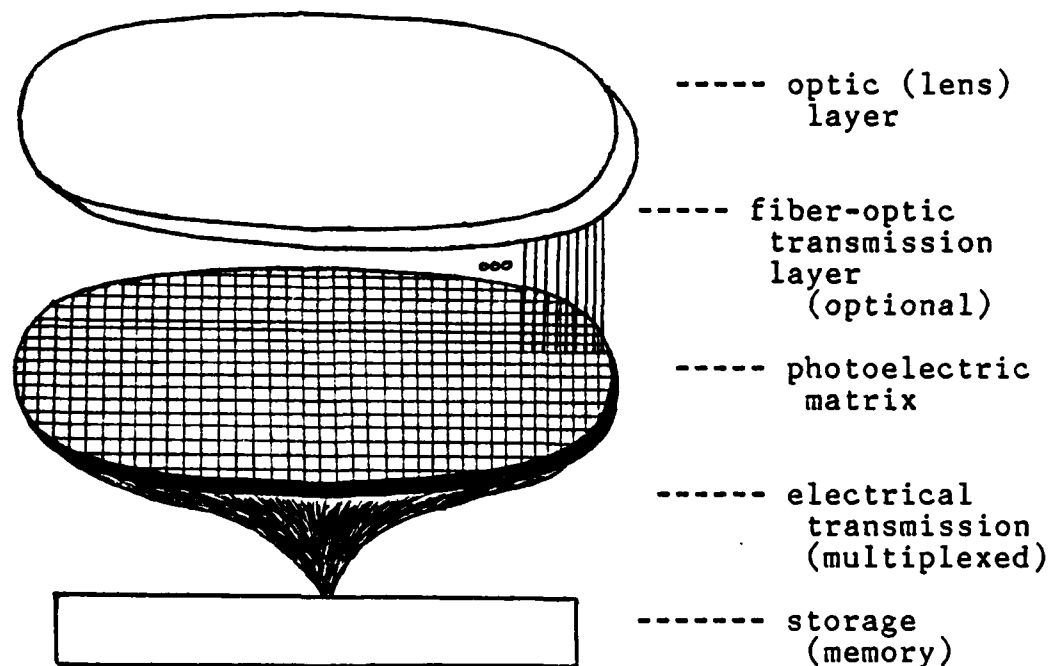


Figure II-6. Concept Diagram of Passive Sensor.

spacecraft could incur unacceptable image errors from the active sensor's scan delay.

A partial solution to this lies with passive sensors: multi-element arrays of detector elements that do not move, similar to the functioning of the eye of an insect. The simplest conceptualization of this would be as an etched mosaic of photo-sensitive material (figure II-6). The use of such a sensor with the extremely weak photon levels of starlight might not have been possible when Campbell's and Potter's papers were published (Ref 2; 17). However the technological state-of-the-art in three areas has progressed substantially since that time: 1) research in medical electronics and general electronic component micro-miniaturization has dropped the threshold of a minimally detectable electrical current by orders of magnitude; 2) the maturation of fiber-optic technology allows the transmission of light with very low levels of attenuation (such as from magnification to detection elements); and 3) constant advances have been made in photo-electronic devices, to the point that even starlight would need little amplification to be usable as input to a gallium arsenide detector element (Ref 3; 22).

The passive sensor would still have some time latency: unless the individual elements can be processed or stored in parallel, a more typical multiplexed input requires a normal sequential processor to read each element in turn, and this

will take a discrete amount of time that is a function of the processor. But the speed of electronics could cut this time substantially from the time delay required with a single element (or multi-element) sensor that requires physical movement.

This project used sensor configurations that are representative of both classes. The primary sensor in phase II and phase III testing was a 35mm camera: essentially a passive sensor. Phase IV testing involved a television camera directly sensing the sky. The TV camera's raster scan was an active function, that produced an image that was then stored and digitized by a separate device (see chapter IV).

Resolution

Two independent aspects of the resolution of the sensor need to be considered in relation to the development of the algorithm. The first concerns the accuracy of the distance measurements. If a sensor resolves an image into N elemental pieces per degree of arc, the possible error of position for a point in the image is bounded by the square root of 2 divided by N ($1.41/N$). Since a distance measurement involves two stars, the maximum measurement error from the resolution limitation would be twice this value ($2.82/N$). For example, using only 6 elements per degree (36 per square degree) would yield an accuracy limit of 0.47 degrees. The algorithm in this project successfully

handled greater inaccuracies than this.

The second aspect of resolution concerns the joining of separate points of light in the image into single stars. Many visual binaries, and some larger multi-star clusters, exist at brightness levels comensurate with those used in this project (as in figures V-1 or V-2). If the resolution is such that the multiple nature of the image cannot be detected, i.e. the images impinge on adjoining sensor elements so that no gap between is detected, then the pre-processing program used will cause them to be treated as a single star. This occurrence can be treated by constructing the database to handle the binary in the same manner as the sensor handles it: either as one or two bodies. A slightly more complex problem results when a binary pair is separated by a distance that is between one and two times the resolution of the sensor. In this case the presence or absence of the actual gap will be determined by the positioning of the sensor, as shown in figure II-7. If this disrupts algorithm performance a solution is to include multiple templates in the database for the same catalog-star when the binary star processing is critical.

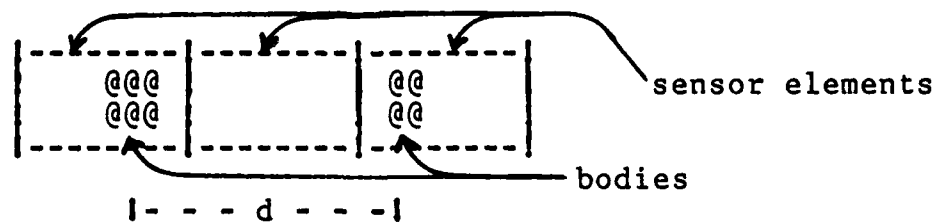


Figure II-7a: Sensor Detection of a Binary Star Gap.

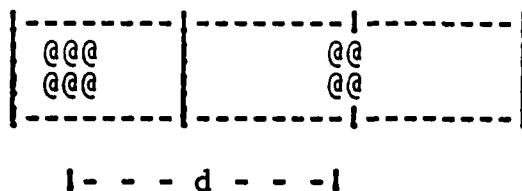


Figure II-7b: Non-detection of the Gap due to Slight
Sensor Repositioning.

Refraction

The refraction of light due to the atmosphere of the earth could distort the apparent distances seen in an earth-surface sensor, but can be essentially negated by avoiding the areas close to the horizon. During standard day atmospheric conditions, above 30 degrees of elevation the refraction correction drops to about one minute of arc or less. Not sensing the areas close to the horizon also avoids the areas of the earth's surface most likely to have sky obstruction and also the area that contains the maximum amount of background light.

Planets

In addition to the true stars, the planets can be used as sources of positioning information. But because of their motions through the celestial sphere, a static template similar to the ones created for the stars is not possible. They are seldom necessary for positioning when the true stars can be detected, but must be dealt with because they act as a disrupting influence (along with meteors, comets, aircraft/spacecraft and any other sky light-source) for any

algorithm utilizing the light patterns in the sensor for body identification. The planets, and the course of action to deal with them, can be divided into three cases based on the brightness of the individual planets.

Venus and Jupiter should be detectable as brighter than the other bodies in the field-of-view, so the algorithm as written will attempt to identify the planet as one of the 57 templates (since it proceeds from the assumption that a catalog star, if present in the FOV, will be the brightest object there) and fail. A solution is to allow the main part of the program to discard the information on the brightest body after a failure (which can be simply done by eliminating the first column of measurements in a sample) and then run it again. How useful this re-running would be would depend on the ratio of processing time to the time required to redirect the sensor and pre-process another image. When the main identification program runs much more quickly, the discard-and-try-again approach would be more effective. When a lesser amount of time is needed to acquire a different image and create another sample from it, simply discarding the failed sample is indicated since there is no guarantee that reprocessing the original image will succeed (due to the possibility of a second planet in the FOV, occultation of a star, et cetera).

At the other planetary extreme, Mercury and the three outer planets are dimmer than all of the catalog-stars, so

the algorithm will not attempt to process them as a catalog star. They might be included in a sample created around an actual catalog-star, thereby disrupting a complete correspondence with that catalog-star's template vector, but since the rest of the normal support-stars would be present the worst effect would be to bump the dimmest of the normal support-stars from the sample list. This could weaken the pattern matching capability somewhat, but the results indicate that a successful match is still an excellent possibility under this condition (chapter V).

Mars and Saturn, if found in the field-of-view, will usually be the brightest objects present, so would have the same effects (and require the same solution) as Jupiter and Venus. Occasionally they might traverse the field-of-view of a bright first-magnitude star, and be considered as a support-star, as in the second category above.

The existence of the planets is a phenomenon that argues strongly for the use of a feature vector that is as large as practical. Attempts to use only a small number of support-stars and precisely match simple polygons (such as triangles) could be highly disrupted by transitting planets.

III. EXPERIMENTAL DESIGN

From the preceeding chapter it can be seen that any star pattern recognition algorithm must contend with a non-uniform distribution of the stars that might be used to provide a supporting geometry around the cataloged template stars. Additionally, practical limits on sensor power, resolution, and field-of-view, database parameters, and image disruption from refraction, planets, aircraft, and so on, all suggest that any algorithm must be as flexible as possible. A device that must function outside of a laboratory environment, in the real world of foreseen and unforeseen adverse conditions, should consider fault-tolerant techniques where possible and appropriate.

Fault-tolerant computing always involves some type of redundancy. Redundancy in hardware generally involves multiple identical components, but software fault-tolerance includes algorithms or data that are redundant in the sense that they are means to the same goal as a primary structure, but are in some manner different from that structure (Ref 18:220). This project tried to provide redundant paths to a successful star-identification, consistant with the actual natural patterns of the stars. Two interrelated pattern recognition procedures were devised, and the database was made richer than strict performance considerations would indicate, so that the elimination of some of the identification elements (support-stars) from either the

templates or the samples would still allow a chance of success.

Database Construction

As stated previously, the set of 57 stars catalogued in The Air Almanac was selected as the template list. A variable length template vector was then constructed for each catalog-star (from the set of fourth magnitude or brighter stars that were within ten degrees of arc of each catalog-star), consisting of measurements of the distances between the support-stars or between the catalog-star and each support-star. The measurement process was carried out by hand, using dividers and two star atlases (Ref 4; 12:457-472) via the following steps:

For each catalog-star, do:

1. Given its declination and sidereal hour angle from the Air Almanac, locate it on the appropriate page of an atlas;
- *2. Draw a circle with a ten degree radius around the catalog-star;
3. List, in order of brightness, the stars of the fourth magnitude and brighter (up to nine) within the circle;
4. For each of the stars from step 3, do:
 - *4a. measure the distance to the catalog star;
 - *4b. measure the distances to all of the brighter (previously processed) support-stars.

The steps marked with an asterisk (2, 4a and 4b) normally involved rescaling the basic measurement (from the dividers) to conform with the changing absolute distance scale on different sections of each star chart.

The resulting set of distance measurements, referred to as the template feature "vector", was stored in a matrix format for ease of visualization. A typical entry is shown in figure III-1, and the entire database comprises Appendix B.

51. Altair - 0.9 / aquila alpha	
1= 2.1	% 2.8 (gamma)
2= 8.6 9.1	% 3.4 (delta)
3= 7.9 9.7 7.0	% 3.7-4.4 (eta)
4= 9.7 8.0 16.3 17.5	% 3.8 (sagitta delta)
5= 2.8 4.9 8.0 5.3 12.2	% 3.9 (beta)
6= 9.6 7.7 15.3 17.3 2.0 12.2	% 4.4 (sagitta alpha)
7= 8.9 7.0 14.9 16.7 2.0 11.8 0.7	% 4.4 (sagitta beta)

Figure III-1: Sample Template Feature Vector.

The columns are treated as numbered from zero to N-1, where N is the number of support-stars included. The N rows are numbered as indicated, one per support-star. The zero column contains the distance measurement from the particular support-star to the catalog-star (in the example, to Altair). The other entries contain the distance from support-star i to support-star j (where i is brighter than j) in column i, row j. Half of the matrix is omitted because its data would be redundant. The main diagonal is also omitted, since it contains all zeros (the distance from any star to itself). The right side of the example contains

the magnitude and identity of each included support-star (from Ref 4). Where the support-star is a member of the same constellation as the catalog-star (here, Auriga) the constellation name is not repeated: support-stars 1,2,3 and 5 are also in Auriga, while 4, 6 and 7 are from neighboring Saggita. The brightness value listed of support-star 3 (Aquila eta) is 3.7 to 4.4, identifying this as a variable star. Note that this star could properly be placed anywhere from third in the list to seventh, depending on its current brightness. Since the brightness correlation (discussed below) depends slightly on a correspondence with the ordinal rankings as contained in the templates, a source of error is introduced when Aquila eta is in a dim phase. Because of this error possibility, and a limited ability of simple sensors to precisely differentiate stars by their brightness, the ordinal correlation is weighted much less than computations that use absolute visual magnitudes (chapter V).

Errors in the Database. The distance measurements are recorded to a single decimal place (tenths of a degree of arc). In many cases, even this minimal level of precision is unwarranted by the accuracy of the measurements, since some errors have been detected in the database as large as nine percent. Several factors are responsible for the inaccuracies:

1. The atlas used for the initial measurements was based on the 1950 epoch. The permutation of the

celestial sphere has caused some shifting in the 23 years since, however this is, in general, a very minor effect compared to the following;

2. The measurements were taken by hand, using dividers and a hand-drawn scale. Because of a desire for precision, painstaking care was used during an already tedious task, but the inexactitude of a human performing a manual task under time pressures had some error effect;

3. The largest source of error, however, was probably inaccuracies in the star charts. There is no perfect method to represent a spherical entity on a flat surface, such as paper. The set of charts used (Ref 12) consisted of conformal or polar projections, and seemed to be quite adequate when a catalog-star was in the center of a chart, but when it (or some of its support-stars) straddled the chart boundaries a noticable level of distortion was observed. Attempts to rescale the majority of the initial measurements undoubtedly helped, but the result was still a database with a large degree of relative error.

A different approach to database creation was possible that would have resulted in a much lower inaccuracy factor. This was to use an automated procedure to create the templates, using simple spherical trigonometric calculations and two input files: the first containing the list (with

declination and right ascension) of the desired template stars, and the second file of all the stars in the heavens of the desired minimum brightness level. This approach was not used here because of the time required to create the second file in the computer, but in addition to greatly increased accuracy, once developed it could be used to quickly create alternative databases that varied the inclusion radius or brightness threshold of the support-stars, or the catalog-star list, or all of these. This method is therefore suggested for construction of an operational database, and a pre-existent computer listing (such as the 300,000 stars in the SAO catalog) might be used as the second source file.

There was, however, one valuable side-effect of the inaccuracies that resulted from the manually created data base: it gave a good indication of the inaccuracy that the algorithm could successfully handle, and this provides information on the resolution that must be specified for a sensor. In other words, this project showed that input from a sensor with relatively poor resolution could still be successfully handled by this algorithm.

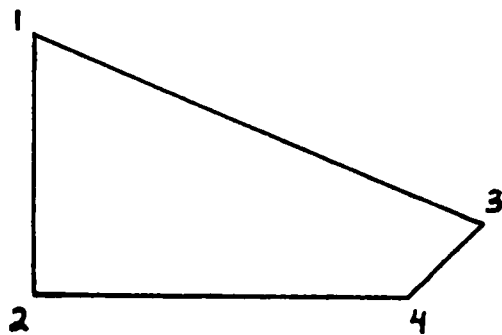
Sample construction is discussed in chapter IV, but both manual and automated sample construction techniques used the same schema as template construction: first determine what light sources (tentatively stars) exist within the specified support radius (ten degrees) of a star that

will be submitted to the main algorithm for possible identification; secondly rank-order these support-stars; and finally extract the Euclidean distances in a matrix format.

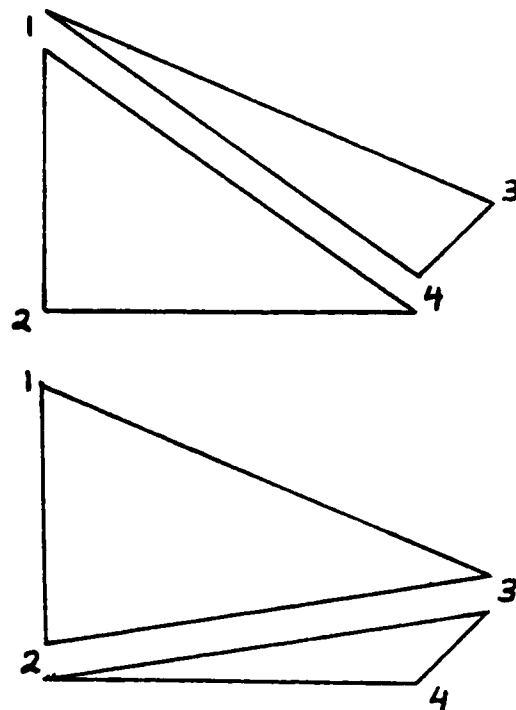
Theory of Nested Recomposition

The database, as constructed, is one method of describing the two-dimensional polygon of $N+1$ vertices formed by each catalog-star and the N selected support-stars. The polygon is decomposed into $N(N+1)/2$ inter-connecting sides, and as discussed previously, this set can be reconstructed to form only two possible (mirror-image) figures. But subsets of each catalog-star's data set can also be used to construct simpler polygons that include the catalog-star as one of the vertices, down to the simplest case of the $N(N-1)/2$ triangles that can be constructed from the catalog-star and all possible pairings of the N support-stars.

These simpler polygons can serve as an intermediate step in the recomposition of a higher-order (greater number of vertices) polygon: a quadrilateral can be constructed from six (consistent) individual side distances (figure III-2a), but also as a two-step process of first constructing triangles and then using a common distance value to join the triangles into the same polygon (figure III-2b). The method of joining is not arbitrary: the identity of the sides must be maintained (such as by retaining the identity of the



(a)



(b)

Figure III-2: Recomposition of a Quadrilateral
from Triangle Pairs.

vertices, as in the example) to prevent an incorrect recomposition from figures that have a side of identical distance out of pure coincidence.

As a pattern recognition application, and specifically from the perspective of the problem in this thesis, a deterministic means to identify a specific polygon composed of a variable number of vertices can therefore follow a sequence of the following steps:

1. Initially construct the set of $N(N-1)/2$ triangles that contain the catalog-star and the possible pairings of N support-stars. These are the entry level polygons for the procedure.

2. Compare these to the set created by the same method from the sample vector, retaining those for which a match can be found between the sample polygon and a polygon from the current template.

2a. If no match can be found there is no common polygon at this level, and the recomposition process halts.

2b. If only one polygon match is found, then it is not possible to construct a higher-order polygon since nothing else exists to be linked with. In this case recomposition also stops, with a successful match at this level.

3. If multiple polygons exist at this level, a linking between them, to create the next level higher-order polygon, is attempted by comparing the unused side measurement in the template set to the same measurement in the sample set. If this distance is the same (i.e. within a specified tolerance level) then the matching is successful: a polygon of the next higher number of vertices (it might or might not be the first one at this level) has been successfully matched between the sample vector and the current template vector. After the supply of polygons at this level is exhausted, the procedure returns to step 2 to attempt to join the new set of higher-order polygons into polygons of an even higher order.

An explanation of the "unused side", as well as the foregoing description in general, should be aided by a basic and abbreviated example.

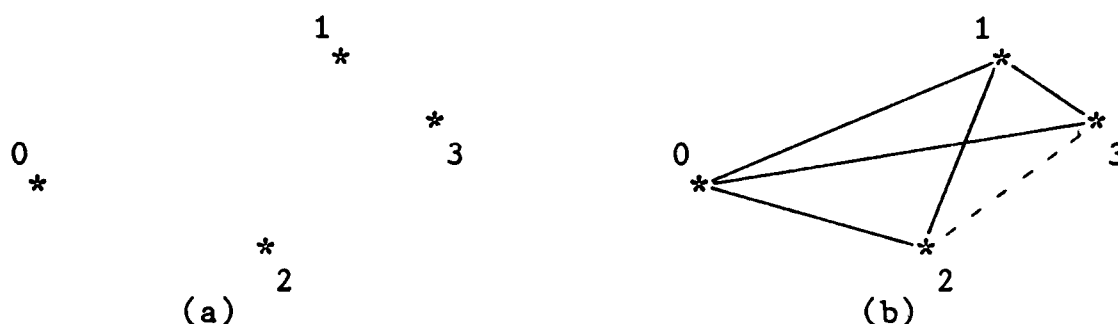


Figure III-3: Recomposition Example.

Given a catalog-star (vertex 0) and three support-stars (vertices 1, 2, and 3) in figure III-3a, the creation of both the sample and template feature vectors is accomplished by decomposing the quadrilateral into the six distance measurements:

0-1
0-2 1-2
0-3 1-3 2-3

This example will assume a perfect sensor: one that processes an image to create a sample that both orders the data and precisely measures the distances identically to the template, so that in all cases an entry in the sample vector (denoted as "S[0-3]" for the sample vector entry in column 0, row 3) will exactly match the corresponding template entry (written as "T[0-3]"). Then following step 1 from page III-8, three triangles are formed from both the sample and template sets as 0-1/0-2/1-2, 0-1/0-3/1-3, and 0-2/0-3/2-3. Since a perfect match is assumed in this example, step 2

requires that all three be retained, i.e. there were three successful pattern matches at the polygon-3 (triangle) level. Step 3 then attempts to link pairs of triangles into quadrilaterals, by comparing the unused side in the sample pairings to that in the template pairings. For the pair of triangles $S[0-1/0-2/1-2]$ and $S[0-1/0-3/1-3]$ shown in figure III-3b, the unused side that could link them together is $S[2-3]$. $S[2-3]$ and $T[2-3]$ always exist (that is, values for them can always be found in the respective feature vectors) but only if the unused side value is the same ($S[2-3]$ equals $T[2-3]$) is there a match that signifies correspondence between the sample and template of a higher-order polygon. Since in this example $S[2-3]$ does equal $T[2-3]$ a match between this sample and the current template at the polygon-4 level has been established.

One way to view this scheme is to note that six-vertex polygons are a subset of the five-vertex polygons that can be linked together; similarly the set of pentagons is a subset of quadrilaterals is a subset of triangles, which are themselves a subset of the individual sides that exist between each pair of vertices. In general, any high order polygons are nested within increasingly larger sets of increasingly simpler geometric structures. The process of creating the templates and samples is a decomposition of a high order polygon directly into its constituent sides in a single step. But the pattern matching procedure is a

process of nested recomposition of simple structures into more complex structures, until all of the template (or sample) vertices are included (a perfect match) or disagreement of the "unused side" values is detected, indicating a partially or completely failed match.

Although this method was developed to handle a geometric feature set, it actually operates on a set of scalar numbers. The computer is unaware of any geometric basis for the algorithm, so this process should be applicable to a general category of pattern recognition problems where the dimension (size) of the feature vector for either the sample or the template (or both) is a non-constant value, or cannot be specified beforehand. For nested recomposition to work, however, there must be some specifiable relationship between the elements of the feature vector. (In the geometric interpretation, the elements compose a self-consistent polygon).

The identification algorithm developed for the computer (Appendix A-1) used the nested recomposition scheme on pages III-8 and III-9 with this modification of step 1: rather than create all of the catalog-star inclusive triangles in both the sample and a current template (and then compare them en masse) only triangles that already were "matched" were actually created. The program started with a more basic geometric unit, the individual sides, and required that each appropriate sample side match the corresponding

value of a template side as the triangles were constructed. This is consistent with the philosophy of nested recomposition, and in fact extends it to include triangles as an intermediate structure, with each individual distance measurement as the entry-level "polygon" that is formed by two vertices. This point was omitted from the previous, more theoretical description of nested recomposition to make that initial explanation briefer and, hopefully, clearer.

Effects of Limited Accuracy

If it were possible to construct a perfect sensor, that is, one that could process an image with a high degree of precision in total agreement with a corresponding (perfect) template, and the possibility of planets or other pattern-disrupting light sources was removed, then three observations could be made:

1. A very small feature vector could be used, since the consistency and precision of the measurements provided by triangular patterns alone could afford sufficient discrimination from all other possible templates or star patterns. No pattern consisting of three stars (or more) is repeated on the celestial sphere if the measurements are precise enough.
2. The need for only two support-stars (if triangles are sufficient for discrimination) removes the difficulties of sparse-field catalog-stars.
3. Nested recomposition is then unnecessary (although

it could still be used in a single step application to pattern-match the triangles) because no match above the polygon-3 level is attempted.

However, the inaccuracies encountered forced the development of a system to handle them. The result of this is probably good: if the processing functions of a celestial recognition system are developed to handle a level of inaccuracy that is higher than would occur in actual operation, it should have no difficulty functioning in a more accurate environment.

The immediate effect of the inaccuracy of the distance measurements was to force a "trade-off" between Type I and Type II errors. The null hypothesis used here is that, given some measurement tolerance, a Menkowski-one (or some other) distance value (from a geometric template structure to the corresponding sample structure) that is less than the tolerance occurs because the current template is a correct match of the sample. In other words, this null hypothesis says that a match occurs because it should: because the sample and the current template represent the same star. A Type I error (an incorrect rejection of the null hypothesis) would then occur when the sample and template do represent the same star, but the computed distance exceeds the specified tolerance, so a "no match" decision is reached. A Type II error (incorrect acceptance) would be a sample-template distance that is less than the threshold value (a

"match"), but between a sample and a template that did not represent the same star. (When single vector elements are compared, for example the comparison of a sample single-side to a template single-side, all of the Menkowski distances equal Menkowski-one). The threshold level, therefore, can be used to control the "trade-off" between the frequency of Type I and Type II errors. But the term "trade-off" is misleading in this discussion, since the frequency of Type I errors should probably be zero: the probability of rejecting a match between a sample structure and the corresponding template structure from the same star, because of a low tolerance for measurement error, should be statistically negligible. For this reason the threshold was set at a "generous" level: the minimum level such that no Type I error (inadvertent rejection of the template that actually matches an input sample) should occur. The initial tolerance level was set to 10 percent of the average measurement (but this was slightly modified during program testing phases: chapter V). The result of totally eliminating Type I errors was to allow a frequent occurrence of Type II errors: incorrect "matches" of a sample to the wrong template because the measurement difference was within tolerance.

The frequency of such a coincidental match can be generally predicted for each level of geometric structure. At a ten percent tolerance level, a uniform random

distribution of distance values will coincidentally "match" a single side at a 0.10 rate. A triangle would cube this figure, to 0.001. But there is a large number of potential triangle (polygon-3) matches: in the maximum case of a ten star sample (nine support-stars) and a rich-field template (also nine support-stars) there are 36 triangles in both the sample and the template that contain the candidate (or catalog) star, giving 1296 (36 x 36) comparisons. At the ten percent tolerance level, therefore, a maximum-entry sample will match an average of 1.296 triangles in a maximum-element template from pure chance. However, an analysis of the next higher geometric level (polygon-4) matching, following the nested recomposition process, shows that the probability of a random match drops substantially.

Since nested recomposition must have two or more triangles to join, determining the probability of a chance polygon-4 match can start by using a binomial distribution: letting N represent the number of support-stars in the current template, and M+1 be the total number of stars in the sample, the number of attempted matches at the triangle level ("T") equals

$$(N(N-1)/2) (M(M-1)/2) \text{ or } (N^2 M^2 - NM^2 - N^2 M + NM)/4.$$

Since the probability of a single match ("p") equals 0.10 cubed, or 0.001, the total probability of two or more triangle matches ("P") is

$$\begin{aligned}
 P == \text{Prob}(n > 2) &= 1 - (\text{Prob}(n=0) + \text{Prob}(n=1)) = \\
 &1 - \left[\binom{T}{0} p^0 (1-p)^{T-0} + \binom{T}{1} p^1 (1-p)^{T-1} \right] = \\
 &1 - \left[0.999^T + 0.001(T)(0.999)^{T-1} \right]. \quad [2]
 \end{aligned}$$

For a maximum size sample compared to a maximum size template (as used in this project) $M=N=9$, which gives $T=1296$ (as above) and $P=0.03718$. These triangles can only be joined with another sample-template match, of the unused side, so this number is multiplied once more by p , giving 0.003718 . Using nested recomposition one additional level, therefore, reduces the Type II error probability by a factor of about 350. For a small sample (e.g. 6 total stars, or $M=5$) and a minimal template ($N=3$) T drops to 30 and P to 0.000004 . Despite differences between the tolerance levels actually used in the program and the ten percent value used here, and the fact that the determination of whether a difference was within tolerance was never as simple as a single value (10%) comparison, the above analysis is in good general agreement with the empirical results. A coincidental match (Type II error) occurred frequently at the polygon-3 level, necessitating continued processing to choose between the templates. An incorrect match at the polygon-4 level, however, was a comparatively rare occurrence that was only observed between a relatively large sample and a rich-field template.

IV. PROGRAM CONSTRUCTION

Three separate computer programs were developed for star identification: they are included in Appendix A. The main program, "Whoswhointheheavens", performs the pattern-recognition function. It was written in Pascal on the Air Force Institute of Technology Scientific Support Computer, a VAX 11/780 using the University of California at Berkeley Unix operating system version 4.1. The second program, "frontendsampleconstructioninputtopatternrecognizer", processes digitalized image data to create a sample for the main program. It was also written in Pascal on the same computer system. The third program, "Video", is a utility to transfer image data from a Colorado Video Corporation Model 280 Video Transceiver as prescribed by its reference manuals (Ref 9 and 10). It was written in the language "C" on a Cromemco Z-2D System Two microcomputer operating under the Cromemco Disk Operating System version 02.36 and CP/M. (A fourth program was written, not to identify stars, but as an improved method to handle the measurements taken from the now-identified stars to resolve a position on the earth's surface. The improvement is that it does not require azimuth information, only the elevation of multiple bodies. It was also written in Pascal on the Vax 11/780, and is included as Appendix D.)

The literal names of important program variables are denoted in the following sections by single quotes.

The Pattern-Recognition Program

Previous chapters have discussed the use of geometry as a star-identification feature set and the use of star brightness as an organization principle. These two aspects separately are used to establish a correspondence between an input sample and the template set, but they are used together to determine the logical flow-of-control in the program. Figure IV-1 shows the overall processing sequence. The program is currently set up to try to match a sample as given; that is, it assumes the sample's candidate-star is one of the catalog-stars (templates), and it attempts to determine which one. It does not currently modify the sample, so if a non-template body (such as a planet, as discussed in chapter II) is the candidate-star, the program will indicate the lack of a good match and halt after trying all of the templates. The program may, however, be easily modified to remove the zero column from the sample (thereby eliminating the brightest body from the sample and substituting the next brightest as the new candidate-star) and run again. Depending on mission requirements, this may or may not be a desirable modification (see chapter II).

After reading the sample data (procedure 'readsample'; figure IV-1a, block 1) the program will attempt to match this data to the set of templates. Each template in turn is read from an external file ('starbase', which is Appendix B) by procedure 'nexttemplate' (figure IV-1a, block 2). Then a

support-star from the sample (block 3) is considered to be a possible match ('hypothesis') to each support-star in the current template ('currenttemplate') in turn (block 4). If the hypothetical matching fails, i.e. if the simple Menkowski-one (M-1) distance from

a: the sample candidate-star to the current sample support-star ('supportsample'),

to

b: the template catalog-star to each template support-star ('supporttemplate') in turn

exceeds the specified tolerance parameters (discussed below), the next template support-star is chosen as the 'hypothesis' and the process repeats. If, however, the M-1 distance is within tolerance, the remaining eligible sides are compared, in the same manner, in an attempt to find a triangle-level match that is within tolerance (blocks 4 and 5). Failure at this level again returns to use the next template support-star in block 3. Success here, however, means that the "first hurdle" has been cleared by the current template: at least one triangle has been found that matches, within tolerance, a consistent subset of the sample feature vector. This makes the template eligible for continued processing by both a polygon-4 construction module 'candidate' (attempted construction of higher-order polygons) and a brightness-correlator module 'brightnesscorrelation'. Goodness-of-match "points" can be acquired by the template from either procedure.

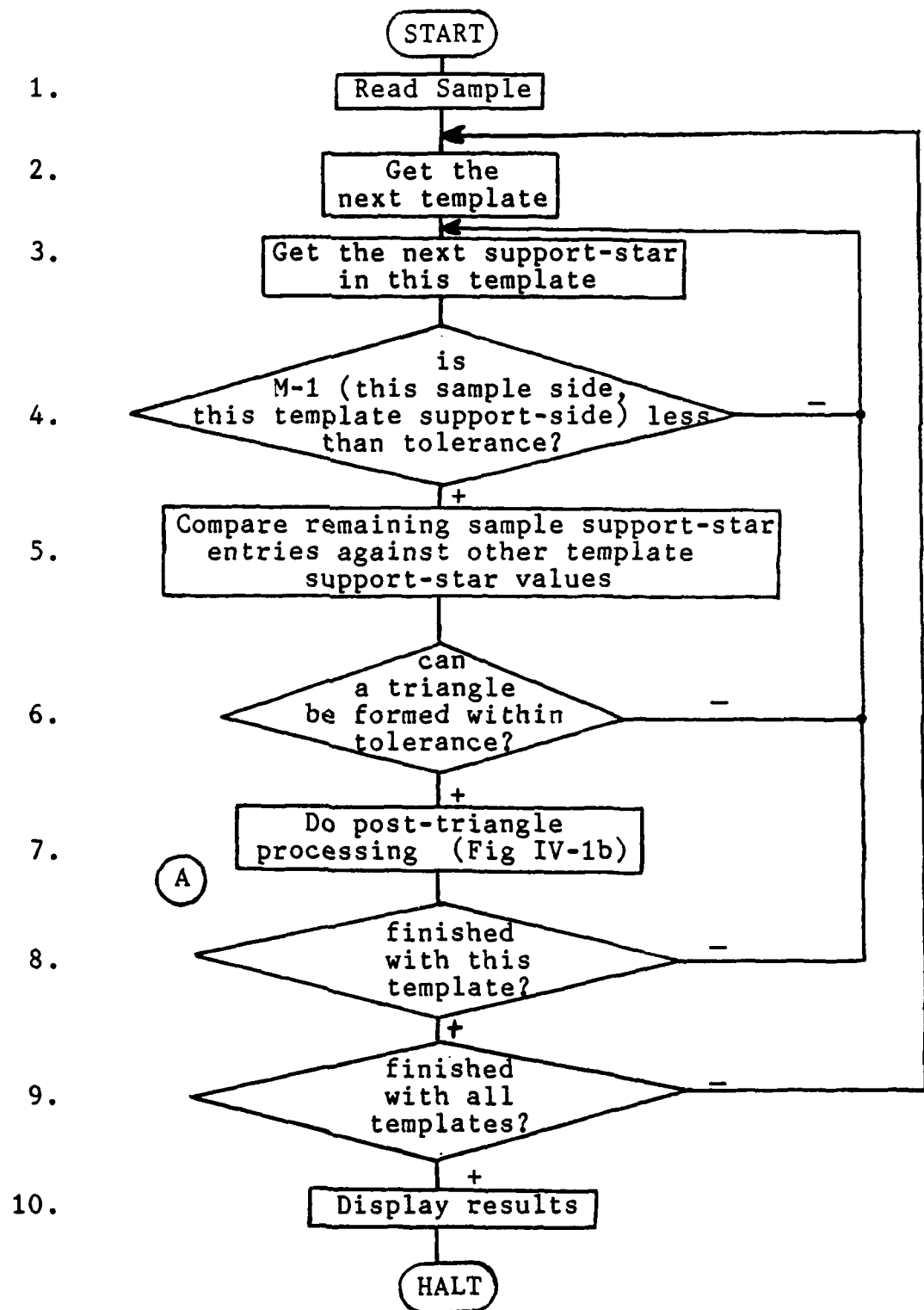


Figure IV-1a: Primary Flow of Pattern Recognizer Processing.

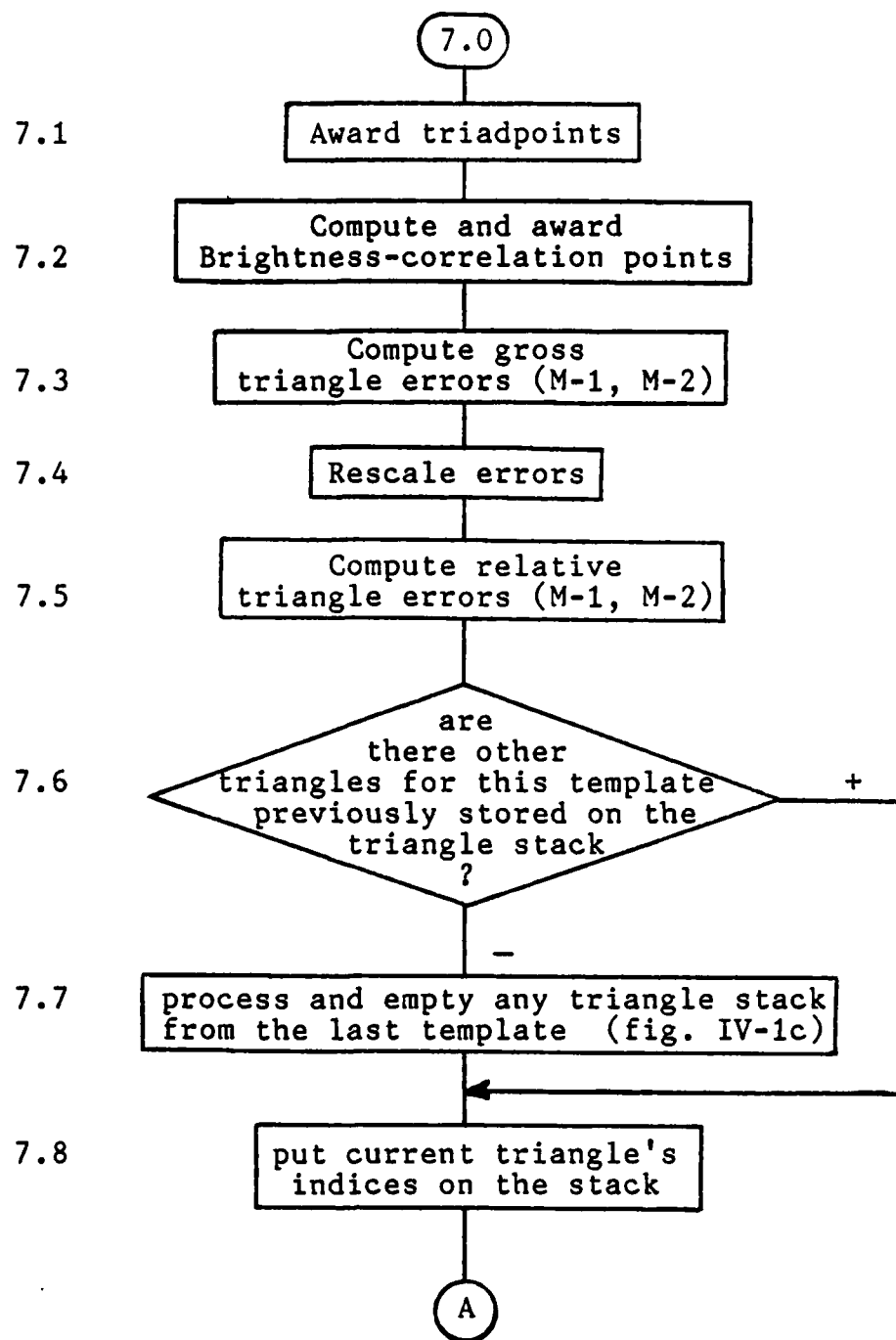


Figure IV-1b: Post-Triangle (Polygon-3) Processing.

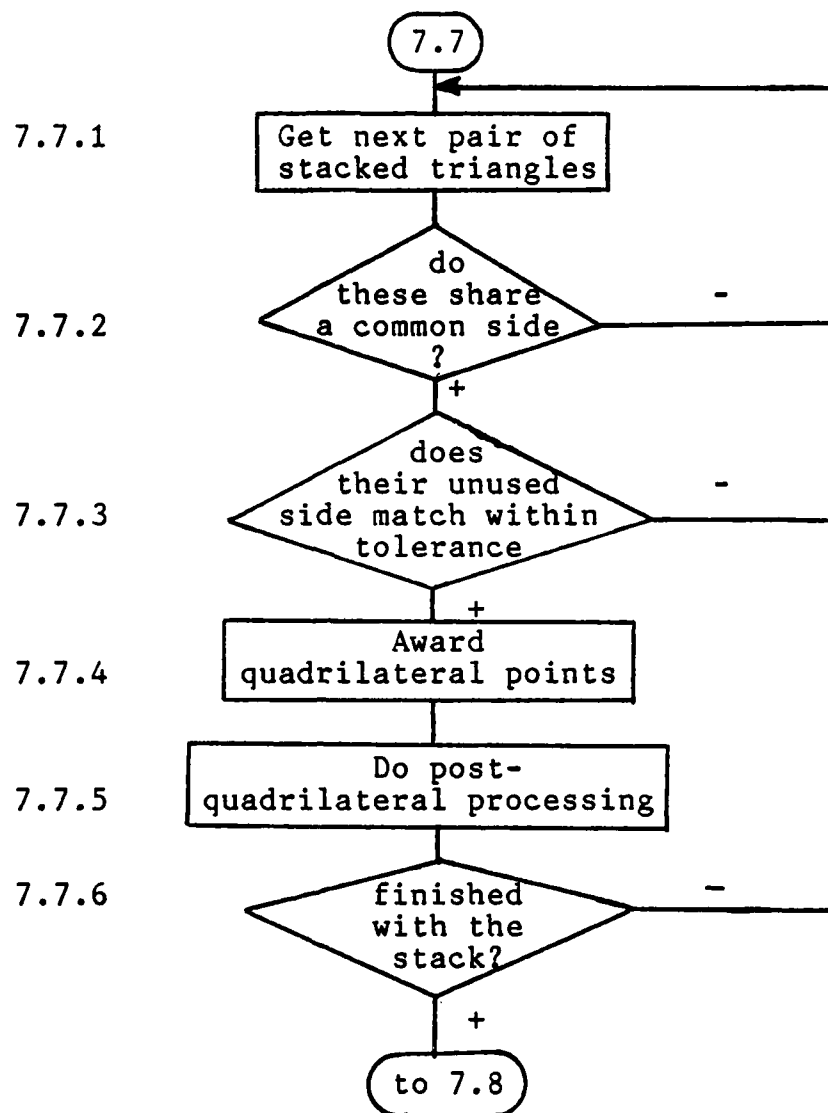


Figure IV-1c: Polygon-4 Level Pattern Recognition Processing.

Post-Triangle Processing. To facilitate program execution, no decision-support processing is started until at least a polygon-3 level match is established for a particular template. The actual decision process is based on the awarding of points to templates that pass this triangle level hurdle. A number of criteria could be used

to select the "best" correlation between the sample and the templates, including:

1. The total number of triangles that each template matches to the sample;
2. The simple Menkowski-one (M-1) goodness-of-fit of a match;
3. The Menkowski-two (M-2) or "Euclidean" goodness-of-fit;
4. A normalized M-1 or M-2;
5. A comparison of the order of the support-stars used in the template triangle to the order of the support-stars that were matched in the sample (ordinal-ordinal correlation). Since the order of inclusion is based on the brightness of the bodies, this is one brightness-correlation method;
6. Comparing the actual stored brightness levels of the support-stars used in the template triangle to the detected brightness levels in the sample (value-value correlations);
7. Variations on 5 and 6 for ordinal-value or value-ordinal brightness correlation;
8. Continuing recursively nested recomposition to join the triangles into higher polygons.

This last approach is considered the best single solution, and it works quite well in most cases (as seen in chapter V). But two contingencies must be considered: no match at the polygon-4 (quadrilateral) level occurs between the

sample and the correct template, or that a polygon-4 match does occur but with an incorrect template. The other seven options can be used to try to cope with these possibilities. Their relative value is discussed in chapter V, but in the final validation phase all the options except 7 and 8 were given zero weight. But since this is an exploratory thesis that tried to maintain a high level of generality in the pattern-recognition aspects of the problem, the mechanisms to use these options are retained even when the weighting assigned to them is zero.

Specifically, as can be seen in figure IV-1b, block 7.1 adds the number of points ('triadpoints') given for successfully generating a match at the polygon-3 level, to the running sum of points (array 'points') for that template. A brightness correlation procedure, discussed in chapter V, optionally adds points through a sample-ordinal support-star to template-value support-star process (block 7.2). Blocks 7.3 through 7.5 show the computation of the gross and normalized Menkowski-one and two distances at the polygon-3 level. The normalization is achieved by dividing each gross error measurement by the average of the two distances compared. For example, if one of the side matches was of a template distance of 8.3 degrees and a corresponding sample distance of 8.7 degrees, the simple M-1 for this single side is 0.4, and the normalized value is $0.4 / ((8.3 + 8.7) / 2)$ or 0.047. At the polygon-3 level, the

maximum M-1 total that would still have allowed a triangle to be considered a match would be $(3 \times \text{tolerance} \times 2 \times \text{support-radius})$ for the unscaled case and $(3 \times \text{tolerance})$ for the scaled case. Maximum M-2 levels use 1.732 (the square root of 3) in place of the 3 in both cases. Points could then be awarded on a linear (if desired) scale using any of these four distance measurements by simply computing

$$\text{'points'} = \text{maximum-goodness-of-fit-points-weighting-factor} \times (\text{maximum-possible-error} - \text{actual-distance-measurement}).$$

Notice, however, that even if the weighting factor used is zero, the distance rule employed still has an effect on the points that a template accrues because it controls the threshold level for a successful polygon-3 level match, without which no points are awarded.

Blocks 7.6 to 7.8 contain a description of the steps used to attempt polygon-4 level matches of the triangles. Block 7.8 will place the currently computed triangle match on a stack-like structure that retains the identities of the vertices ('geomkey'). But first the stack is checked for previously stored triangle matches that could possibly be joined to the current triangle. If others exist on the stack, the current one is simply added to it and program flow returns to attempt to form another triangle. If, however, the previously stacked triangles cannot be linked to the current one, the stack is processed (block 7.7) to attempt to link those prior triangles with each other. This block is expanded into figure IV-1c. Each pair of stacked

triangles (block 7.7.1) is checked for the pre-condition of a common side (block 7.7.2) as indicated by the stored vertices, and then a check that the unused-side template and sample distance values are within tolerance of each other is performed (block 7.7.3). The points for a successful polygon-4 level match are added to the template's running sum if successful (block 7.7.4) and the stack entries and control variables are reorganized to prevent incorrect re-use of the structures.

In all the preceding discussion the tolerance function ('tolerance') used to determine if a template value "matched" a sample value was referred to as a simple comparison of a normalized Menkowski-one distance to a pre-specified constant. Throughout much of the first phase of program testing, however, higher order distance rules and different numeric hurdles were installed to test the effect of different distance rules on the success rate of pattern matching at (primarily) the polygon-3 level. In the case of single value comparisons, where all the higher Menkowski distances are equal to M-1, exponential distance rules were tried to approximate the penalty that M-2 or M-3 attaches to widely variant values ("outliers"). However, this was empirical testing that had little theoretical basis, and no complex or elaborate scheme was found that made a consistent improvement over using a direct percentage of measurement (normalized M-1). This by itself, however, is somewhat too simple since it unfairly discriminates against small

absolute distances, due to human limitations in constructing the database. For example, using a specified tolerance level of ten percent of the average distances being compared would allow a 9.5 degree sample measurement to match any template measurement from 8.6 to 10.5. However, a 1.8 degree measurement would match values from 1.7 to 1.9, and this amount of absolute accuracy was difficult using the dividers and star atlases. Therefore both a minimum acceptable tolerance (the final value was 0.5 degrees) as well as the relative tolerance level (the final, empirically chosen value used for this was 9.4 percent) were used. In the example above, the domain for matching the 9.5 degree measurement would be the same, but the 1.8 degree measurement would now match any value from 1.3 to 2.3 degrees.

Outputs from the Pattern Recognition Program. Each template that passes the triangle match hurdle receives some points, either directly ('triadpoints') or from a polygon-4 match (if present) or from the brightness correlation procedure. In order to accumulate statistics on the total point levels of both correctly identified samples and coincidental point accumulations from Type II errors, the program can output the point values (broken down by category or totaled) for every template that receives points during a particular run. The display of this data is suppressed in the "cleaner" demonstration edition of the program that is

in Appendix A-1, which simply prints out the name and catalog number of the star that has been identified, and a "confidence rating" of the value of this particular result. The three possible ratings of "high", "moderate", or "low" are not rigorous statistical statements, but the confidence rating "high" is intended to indicate at least a 98 percent chance of a correct result, with "moderate" indicating the given answer will correctly identify the true celestial body in 85 to 98 percent of the cases. A "low" rating is the default case, and indicates that the star suggested by the algorithm stands at least a 15 percent chance of being in error. The rationale for these levels is covered in the next chapter.

Front End Processing

The second program, in Appendix A-2, was constructed to create a sample in the format expected by the main pattern-recognition program from pixel data created by the Colorado Video image digitizer. The input to this program was a file ('pixels') stored as 32K (or 64K) of characters that represent the brightness levels of each point in a 128 by 256 (or 256 by 256) matrix. The program was organized around three modules. The first module ('getpoints') compared each of the 32,768 points with a threshold brightness level and recorded the coordinates of each point that was above the threshold level. (If an insufficient number of points is found to allow normal processing, the

procedure decrements the threshold level and begins again). A typical result of this module (using a sparse-field candidate-star) is shown as figure IV-2a. The second module ('joinspots') re-formed the original images by joining the appropriate spots, as indicated by the coordinate data, into "stars". Its output, from the same sample as figure IV-2a, is shown in figure IV-2b. The final module then computes the Euclidean distance between the primary stars, with the result as shown in figure IV-2c.

SPOT NUMBER	X-COORDINATE	Y-COORDINATE	BRIGHTNESS*
0	43	60	1
1	51	48	1
2	54	51	2
3	66	103	6
4	66	104	7
5	67	102	1
6	67	103	12
7	67	104	14
8	67	105	3
9	68	102	3
10	68	103	12
11	68	104	13
12	68	105	2
13	69	103	11
14	69	104	6
15	70	103	2
16	79	37	2
17	80	37	1
18	96	61	6
19	97	61	5
20	100	42	2
21	104	199	3
22	105	199	3

Figure IV-2a: Output from 'Getpoints' Module.
*Brightness is value above threshold.

SAMPLE STAR #	X-COORDINATE	Y-COORDINATE	TOTAL BRIGHTNESS
1	43	60	1
2	51	48	1
3	54	51	2
4	67	104	92
5	79	37	3
6	96	61	11
7	100	42	2
8	105	199	6

Figure IV-2b: Output from 'Joinspots' Module.

from/to	4	6	8	5	3	7	1
6	4.987						
8	9.838	13.30					
5	6.545	2.828	15.78				
3	5.247	4.151	15.05	2.755			
7	6.753	1.867	15.10	2.076	4.507		
1	4.819	5.097	14.63	4.108	1.367	5.748	
2	5.600	4.504	15.42	2.893	0.408	4.747	1.387

Figure IV-2c: Output from 'Makesample' Module.

The logic in the first and third modules was straightforward, however the processing required to correctly reform the star images from the component points proved non-trivial. The basic concept was to again place the above-threshold pixel coordinates ('brightspot') on a stack-like structure (array 'heap') until a sufficient change in both coordinates indicated a "gap" (sub-threshold area) had been traversed between the current and previous 'brightspots'. This indicated that the previous image had ended, so the stack was processed and emptied. Since the stack that had been created might have contained multiple bodies, the

procedure could recursively look for additional gaps that were internal to the stack and create increasingly smaller sub-stacks. The output of this procedure was a reduced set of coordinates, that retained for each star the point where it reached maximum detected brightness and added the brightness values for the other points (that were part of the same body) to the original value for this maximum brightness point. The coordinates for these other points were then removed from the list, so that the final module needed to compute only the distances from the single points that represented the "center" of each star.

Video Digitizer Driver

The final program in Appendix A was used to strobe the data out of the Colorado Video Corporation image digitizer and store it on a magnetic diskette. The program followed a repeated sequence of

1. Send out an x coordinate with the appropriate control signals (in accordance with Ref 10);
2. Send out a y coordinate, again with control;
3. Read the brightness level that corresponds to this x/y address pair.

Other than some timing problems, this program was very straightforward. A more complete description of the hardware that was constructed and used for this processing phase may be found in Reference 16.

V. TESTING, VALIDATION, AND RESULTS

The testing of this project proceeded in four phases. Phase I took place during the main program construction period and had the purpose of verifying the expected functioning of the sub-programs as they were developed. This phase used artificial data that were always "perfect": the input sample exactly matched an entire or partial template that was stored in the database. Except for its usefulness as a cross-correlation check (determining how one template would fare in an attempted match against all of the other templates) it did not provide much insight into system performance because it circumvented the inaccuracies that were encountered, and the far-reaching effects of the system contending with the inaccuracies. Phase I is therefore not discussed further.

The second testing phase used "real" data extracted from 35 millimeter photographs of the sky. The photographs were hand processed in a manner similar to that used during database construction. The resulting sample vector was composed of a sequence of star-pair distances ordered on the basis of the apparent relative brightness of the stars in the photograph. The results of phase II testing are discussed in the next section.

The third phase of testing removed the human element from the image processing. The 35 mm photographs were still used to capture the sky images, but the hand processing of

these images was replaced by the Colorado Video image digitizer, the Cromemco microcomputer, a general purpose television camera (Dage model Six-Fifty) and the two additional computer programs (Appendix A-2 and A-3). This phase is discussed in the second section.

The final phase of testing eliminated the photographs and used the television camera "live": the stars were sensed directly from the camera positioned on the roof of building 640. However, hardware faults and limitations prevented much useful data from being acquired during this phase. Specifically, the camera was limited in its low light-level sensitivity and the video digitizer was operating in a degraded mode. A faulty chip or connection, that could not be corrected after extended trouble-shooting, prevented the extraction of one half of the data (that represented the bottom half of the digitized image). This required that the television camera field-of-view be expanded so that the image of interest was reduced to occupy only the center portion of the top one-half of a CRT monitor screen. The camera has the capability to provide resolution and light detection equivalent to the human visual system, which is more than sufficient for this recognition system, but when forced to allow a field-of-view large enough to compensate for the digitizer fault, these levels are reduced enough to preclude a useful amount of data for most of the sky segments. The phase IV testing, therefore, added little

to system validation.

Phase II Testing

Most of the analysis of the system's performance comes from the second testing phase, which actually consisted of two sub-phases (called IIA and IIB in this discussion) that used the same manual sample creation process on different sample sets at different times. The primary difference between them was that phase IIA testing attempted originally to use a geometric feature set and nested recomposition as the exclusive pattern recognition mechanism. A secondary difference was that the IIA sample set was reprocessed many times as the system "knobs" were tuned; i.e. while the values assigned to the main decision parameters, listed in table V-1, were adjusted to achieve the best results.

The first major observation to surface during phase two testing was that rich-field stars were identified quite well using nested recomposition to only the polygon-4 level. Five of the eight stars in the IIA sample set corresponded to templates that had five or more support-stars. All of these were correctly identified, and the point ratio of the correct template to the most successful incorrect template (the catalog-star with the next highest number of points) was generally 2.5 to 1. (The actual numbers varied with the parameter settings used for a particular run. The figures for both the IIA and IIB sample sets, using the final parameter settings, are in table V-2).

Table V-1: Primary Program Decision Control Variables.

PROGRAM IDENTIFIER	USE	FINAL VALUE	JUSTIFICATION
M: quadpoints	award for a match at the polygon-4 level	100	A: number selected to base other award values upon
M: triadpoints	award for a match at the polygon-3 level	0	E: some points still likely through brightpoints
M: brightpoints	overall weighting for brightness correlation results	30	E: set for maximum help of sparse-field templates
M: weighthypo	award for brightest support-star used	2.0	A: basis for brightness weights
M: weighttemp	award for dimmer support-star used at polygon-3 level	1.0	I: half of the importance of brightest star
M: weightordinal	award for rank order similarity of support-stars	0.01	E: much less use than brightness value comparisons
M: tolerance*	allowable error for a distance "match"	0.094 (9.4%)	E: set to minimize Type II error rate
F: baselevel	remove background light from video image after capture	A/R	Self-adjusting

KEY: M : identifier from main (pattern-recognition) program.
 F : identifier from sample construction (frontend) program.
 E : final value setting of this parameter based primarily on Empirical testing and observation.
 I : final value of parameter mostly from Intuitive reasons.
 A : this value is an Arbitrary base for other parameters.
 * : actually set through 'epsilon' as tolerance = 2/epsilon.

A typical example of the five samples in the rich-field category is shown in figure V-1. Figure V-1a shows the photograph, with figure V-1b a skeletal diagram to show the make-up of the sample in part c.

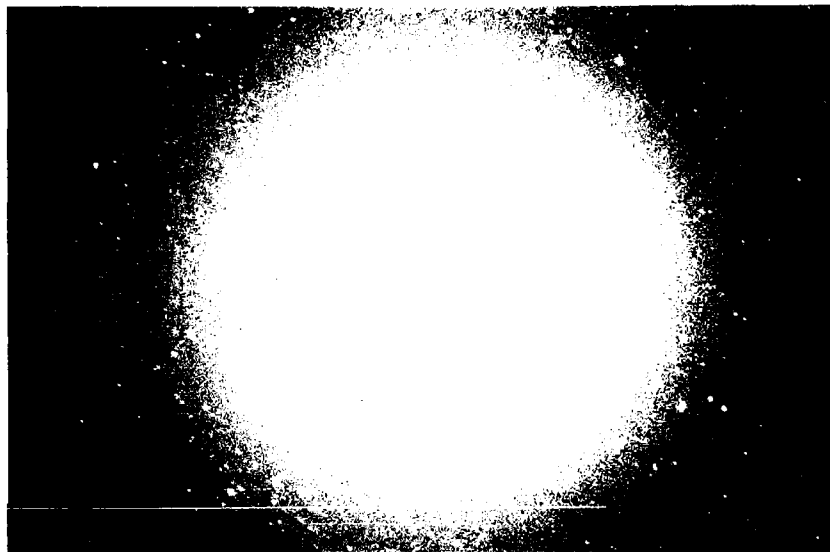


Figure V-1a: A Sample from Testing Phase IIA.

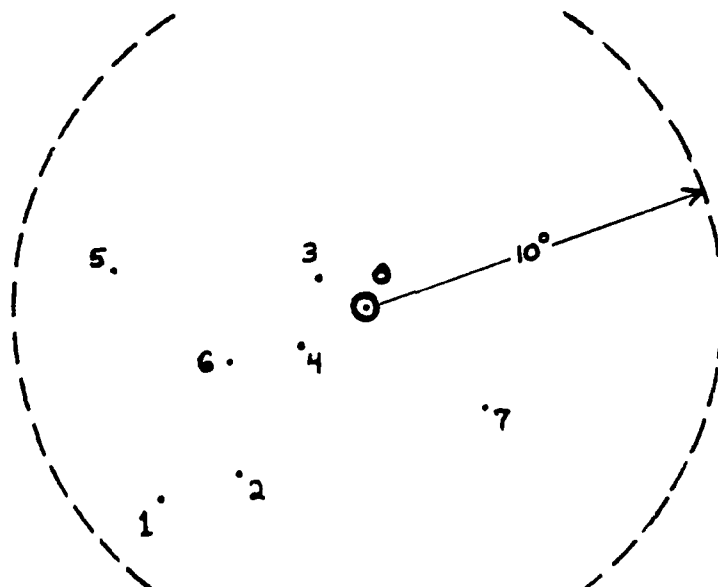


Figure V-1b: Primary Bodies Used in the Resulting Sample.

7.7						
6.1	2.0					
1.8	7.7	6.4				
2.0	5.9	4.4	2.1			
7.1	7.1	7.6	5.7	5.9		
3.9	4.5	3.7	3.4	2.0	4.5	
4.3	8.8	6.9	6.1	5.2	11.1	6.9

Figure V-1c: Sample Data Submitted to the Main Program.

With this sample, template number 49 ("Vega") received 209 total points with the final parameter settings. The next best template achieved 49.6 points. This margin level, or "confidence ratio" made identification of the rich-field stars a robust process that was successful with even wide variations in parameter settings.

In contrast, the sparse-field cases generally had reduced margins of success, and in fact were not totally successful. Figure V-2 shows the same sequence as above,

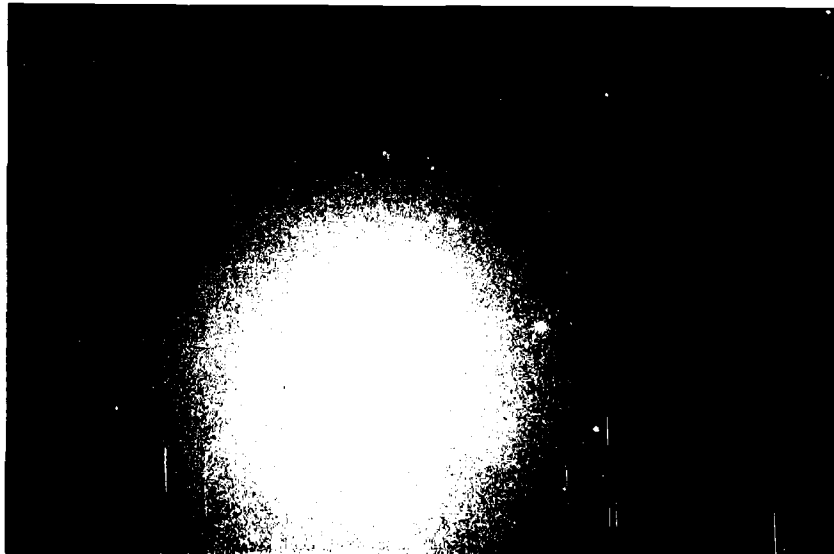


Figure V-2a: A Sparse-field Sample Used in Phase IIA Testing.

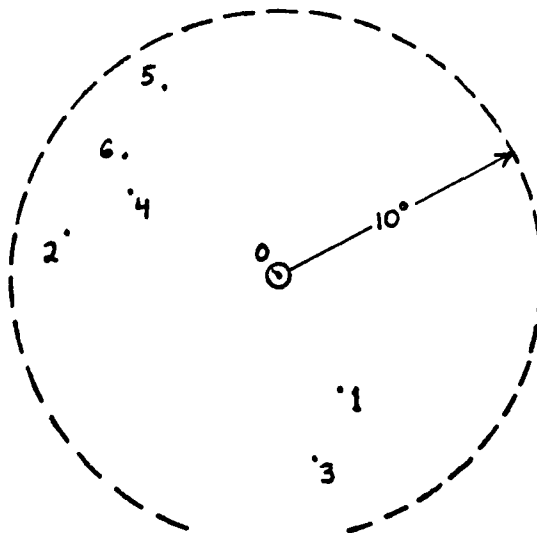


Figure V-2b: Primary Bodies Used from This Photograph.

5.0					
8.0	12.1				
6.7	7.6	6.6			
6.6	11.1	2.8	7.2		
8.3	13.3	5.8	10.9	3.8	
7.2	12.1	3.3	8.5	1.2	2.7

Figure V-2c: Resulting Sample Submitted to Main Program.

with the bright body in the photograph "Arcturus" (template number 37). The program mis-identified this star as template number 05 (Achernar). This selection was the result of Type II errors at the polygon-3 level that were possible because of the high match-tolerance level, which was necessitated by the sample and database inaccuracies, as discussed in chapter III.

Parameter adjustments were therefore directed at the best differentiation of the sparse-field cases. However, no reasonable set of parameter values was created that allowed successful identification of all of the stars in the sample set. For this reason a second feature set was added to the

nested reconstruction processing of purely geometric data. This second method used the brightness of the stars as a possible addition to strict geometry. A detailed discussion of the brightness correlation has been deferred until now because this is the point in system development when it was added. The brightness correlation scheme is basically an opportunistic retro-fit method intended to help identify sparse-field catalog-stars. It is opportunistic in that it uses data that is already available: the relative brightness of the stars was already being used, but as a method of organizing the processing (as discussed in chapter II) rather than as a feature set. It was retro-fit to a program that was successfully handling 75 percent of the sample cases using only geometric features through nested recomposition. Its development was motivated by two factors: 1) the error level in the database, and 2) the use of a pre-existent set of catalogued stars as the data base templates that were included without any consideration of their suitability for automatic identification.

Chapter IV listed four methods of correlating sample brightness data with template data. Two of these cases required that the sensor supply absolute levels of detected brightness to the program, which might be an excessive requirement for either a simple sensor or a sensor on the surface of the earth that must contend with atmospheric variations. However, no such restriction exists on the template side: the brightness values are already specified

as absolute visual magnitudes. Therefore relative sample brightness levels can be compared to either relative (ordinal) template levels or absolute template levels. To maintain generality, both of these were used, but both intuition and empirical results indicate that using the absolute values already stored in the database is a more powerful discriminator, and it therefore received a much greater final weighting factor.

The brightness correlation procedure was placed in the program flow just after the polygon-3 match level of the overall nested recombination scheme. Waiting until a larger structure match occurred exacerbated the sparse-field problem: they have much less fault-tolerance than the rich-field templates, and in the worst case catalog-stars (those with only three support-stars) the omission of a single support-star by the sensor will prevent any polygon-4 level match of the correct template.

The brightness correlation procedure follows this sequence of steps:

1. The magnitude values as contained in the data base are inverted (so that a larger number now indicates greater brightness) and normalized relative to the dimmest support-star in the template. This is accomplished as each new template is read ('nexttemplate') from the database ('starbase').

2. Because the original magnitude values represent

a logarithmic scale, the resultant values are rescaled again to yield a controllable, approximately exponential, differentiation of brightness values.

3. Points are then awarded, based on the rescaled values of the brightness, for both of the support-stars used for the successful polygon-3 level match. (The third matched vertex of the this triangle is the template catalog-star and sample candidate-star).

4. Points are also awarded for the closeness of an ordinal match, e.g. a match of the fourth-ranked sample support-star with the fourth-listed template support-star will be worth more than matching a third-ranked to an eighth-listed. However, a limited ability for a sensor to differentiate relative brightness levels makes this ordinal-to-ordinal statistic of marginal utility, and it is weighted less than 1 percent of the value of the points in step 3.

The overall effect of the brightness correlation is to award maximum points for a successful triangle level match using the brightest sensor support-star with a bright template support-star, and to withhold maximum points if a template vector indicated that a bright support-star should be available to the catalog-star but no star in the sample can be matched to it.

The procedure forms a non-linear equation set, which is kept general by using different weighting factors for each vertex, the ordinal match, and the total brightness correlation function. Additional checks had to be placed in

the program to ensure that a template did not receive brightness points more than once for the same set of vertices, or if it did, the lesser award was not added to the total template points.

The brightness correlation in general worked: it was a minor aid to the correct identification of the rich-field stars, but usually made a measureable contribution to the program's ability to recognize sparse-field candidates. However, in this testing phase the program never successfully identified the Arcturus sample in figure V-2, although this proved to be the only instance of mis-identification.

Because of the parameter adjustments that occurred throughout phase IIA testing, there was a danger that the program was becoming "tailored" to this specific sample set. Phase IIB testing, therefore, submitted a different sample set (again manually constructed, from a different set of photographs) to the program without any program modifications, and with the brightness correlation now used as an adjunct to nested recomposition. This phase was entirely successful. Rich-field catalog-stars continued to (usually) out perform sparse-field stars in both the total point accumulations and in their "confidence ratio": the best-match point total divided by the second highest point total achieved by all of the other templates. In the best case, this margin was 12.98 to 1 (table V-2, sample 14).

Table V-2: Combined Results from Phase II Testing,
Final Parameter Values.

SAMPLE / STAR NUMBER	NUMBER OF TEMPLATE SUPPORT- STARS	CORRECT ID	POINT TOTAL FOR CORRECT TEMPLATE	HIGHEST INCORRECT TOTAL	CONFIDENCE RATIO ASSIGNED
1/46	5	YES	248	81.4	HIGH
2/03	8	YES	267	35.6	HIGH
3/37	3	NO	59.7	80.1	LOW
4/44	8	YES	263	122	HIGH
5/53	9	YES	273	37.3	HIGH
6/49	8	YES	209	49.6	HIGH
7/32	3	YES	111	82.8	LOW
8/34	3	YES	150	40.9	HIGH
9/57	4	YES	232	48.7	HIGH
10/54	3	YES	133	129	LOW
11/27	3	YES	56.7	35.8	MOD
12/**		**	9.9	62.5	LOW
13/03	8	YES	356	34.8	HIGH
14/40	6	YES	248	19.1	HIGH
15/01	6	YES	303	65.9	HIGH
16/50	8	YES	320	45.6	HIGH
17/48	8	YES	269	93.3	HIGH
18/**		**	94.3	97.1	LOW
19/53	9	YES	131	43.5	MOD

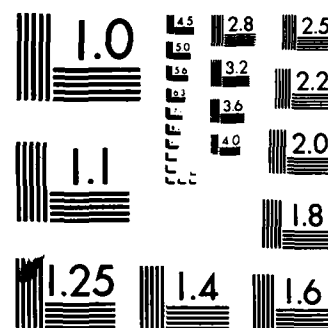
** Sample 12 and 18 were believed to contain a catalog-star (as with all of the samples above) however this proved false for these two cases.

Phase IIA samples are above the dashed line, IIB samples below.

2/2.

NL

END



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Although no mis-identifications occurred in the IIB phase, two instances were ruled as neither correct nor incorrect. All samples were submitted because I "recognized" each photograph's candidate-star as one of the catalog-stars in the database. My original "recognition" was re-evaluated and found false in two instances, however, after the program obligingly pointed out my error.

Phase III Testing

Phase III testing was conducted not to try to improve on the performance of the main program so much as to increase system "objectivity" by removing the human element. More equipment was used and two additional programs were developed to completely automate the system, so this phase primarily paralleled the phase I approach of functional verification. However two occurrences during this phase were significant in terms of overall system performance. The first was that the completely automated system correctly identified the Arcturus example (figure V-2) that phase II testing never conquered. Using the same photograph, the sample created by the pre-processing program proved more correct than the manually constructed sample used in phase II, and the difference was sufficient to allow the main program to successfully identify this sample.

Counterbalancing this success was a diminished ability to pick out "proper" candidate-stars on the first attempt. This was due in small measure to the non-linearities in the

film itself and in the ability of the lens to focus light with perfect uniformity across the film surface, and to a greater degree on hardware limitations. An example of this last effect occurred in the processing of a photograph of the "handle of the big dipper". Two of the stars, Alioth (#32) and Alkaid (#34) were members of the database, but a star between them (Ursa Major zeta) was not. However, this star is a visual binary, and the degraded capability of the video digitizer forced a reduced resolution to a level that eliminated the small gap between the stars.

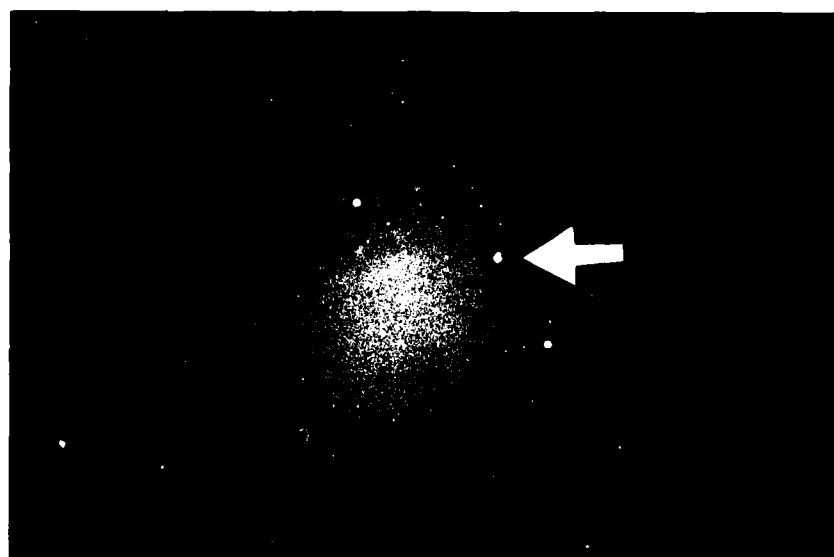


Figure V-3. "Improper" Selection of Candidate Star by Automatic Sample Construction Program. (Arrow indicates Ursa Major zeta).

This caused the 'joinspots' module in the sample construction program to join the two into a single resultant star. Since the stars in Ursa Major are of fairly equal brightness, this caused zeta to become the brightest

eligible body in the image. It was therefore processed as the candidate-star even though it was not itself catalogued in the database, while two bright stars that surrounded it did have templates but were not selected as the candidate.

Results

A figure for the rate of success for this system will not be offered, since any single number would be misleading. A figure of 100 percent could be argued, because all of the samples were correctly identified during phase II or phase III testing. However, phase II failed on Arcturus due to a poor manually constructed sample. Phase III succeeded with Arcturus but failed to choose a "proper" candidate-star from Ursa Major.

A better figure than 100 percent might be the 94 percent success rate from table V-2. Again, this number can be misleading due to the limited number of samples, and to a much larger degree because of the low confidence ratings on some of the matches. The main program classified the results into three categories: high confidence, moderate, or low confidence in the results. These are based on the gross point totals and the confidence ratio (the points of the selected template divided by the points for the next highest template). The highest total points ever achieved by an incorrect template (one that did not correspond to the input sample, whether or not this template was selected) was 129 with the final parameter settings. The criteria for the

confidence ratings used this as a primary benchmark. A "high" rating satisfied any one of three conditions:

1. If the total points exceeded 200, or
2. If the confidence ratio exceeded 3.3 (i.e. the next-highest template point total was less than 30 percent of the selected template's total), or
3. A combination of a point total in excess of 135 AND a confidence ratio greater than 2 to 1.

A "moderate" confidence level was also arbitrarily defined:

1. A point total above 150, or
2. A confidence ratio above 1.5.

If a "successful" identification is defined to require a minimum confidence rating of "moderate", then two of the previously successful samples in table 13 are redefined as failures, and the success rate drops to 82.4 percent.

This figure is still misleading. Since the problems always occurred with sparse-field catalog-stars, and since the database is directly under our control, we can eliminate the sparse-field templates or replace them with rich-field stars, and climb back up towards a 100 percent rate.

Which, finally, is still misleading since it ignores other real considerations discussed earlier, especially the effect of sensor field-of-view. For all but large (45 degree) field-of-view sensors, an arbitrary image formed by pointing the sensor at a random segment of the sky will be processed, but sometimes will be unable to achieve any match

with a reasonable (i.e. high or moderate) confidence rating because of the absence of any of the databased stars in the image. There should be nothing fatal about this "error": another sky segment can be processed immediately.

For these reasons, no single number is an adequate description of the expected rate of success of this system in an isolated real-world instance. However, given the number of catalogued stars that are present in the sky at any time, and that only a few of these need to be identified to allow for determination of position, and finally given the relative level of success of this project in star identification, it is possible to state that the identity of a sufficient number of the available stars can always be established.

VI. CONCLUSIONS AND RECOMMENDATIONS

Suggestions for Improved Algorithmic Performance

Due to the imprecision of the database, a wide margin for error (i.e. a high match tolerance level) was necessary in this project to ensure that a valid ("correct") measurement was not excluded. The effect of this, however, was to allow a great number of templates to qualify at the single-side and polygon-3 levels. A well constructed database, along with a minimally accurate (or better) sensor, could greatly reduce the number of false matches at all levels. This might eliminate completely the need for the secondary pattern-matching algorithm that used support-star brightness correlation as the measured feature.

A second method of database improvement would be to include the same number of stars in each template. Limitations on time and information relating the exact luminosity of fifth, sixth, and seventh magnitude stars in a specific area of the sky led to including only fourth magnitude or brighter stars in the database (in most cases). Enlarging the template base with additional support-stars would incur additional processing time, since the sparse-field stars are currently processed very quickly, but this could remove the relative difficulty that a geometry-only based recognizer had with the sparse-field set that led to the additional development of the brightness correlation scheme. This improvement in the processing aspects would, however, necessitate a more sensitive sensor than that

stipulated here.

A third, major improvement in the database would be to tailor it to the algorithm. For the sake of objectivity this project used as templates the set of stars catalogued by the Naval Observatory. While this set has several strong points, it includes both sparse-field stars and rich-field stars and, in fact, has a preference for the sparse-field stars because the probability of a human navigator observing the wrong star through a sextant is less in an uncluttered star field. Reversing this bias, using rich-field stars for the database whenever possible, should make the recognition system's job much easier.

Numerous methods can be found to increase processing speed since efficiency was, at best, a tertiary consideration in program construction during this project. Even so, processing time was reasonable: the pattern recognition program used about 20 seconds of VAX CPU time for a worst case (largest) sample, and the sample construction execution time was similar. The goals of successful operation, generality, clarity and controllability all were given preference to efficiency when such choices were present.

Performance Summary

The project as described herein did not have the data base advantages listed above. Nonetheless it was successful. Chapter V details the reasons for not using a single statistic as a claim of system performance, but this thesis

was primarily a feasibility study to determine if and how a machine might identify celestial bodies with no input information except the sensor image. Two results are apparent: with the limitations of the database and general purpose sensors (a TV camera and 35 millimeter camera) the system can identify most stars with a high level of confidence using only a geometric feature set and the nested recomposition procedure. This level of success should suffice for navigational requirements for missions in all regimes where mankind currently uses unmanned vehicles, but the recognition capability can be further enhanced, if desired, with a secondary feature set based on brightness correlation.

Recommendations

The Department of the Air Force should build, or contract to have built, a prototype digital celestial self-locating device. Two feasibility studies should then be conducted. The first should determine the possibility of eliminating the navigator crew position (or re-classifying it as an enlisted billet) in multi-engine aircraft due to an increased automatic, cold-start navigation capability. The second study should determine the benefits and costs (power, weight, volume) of including an autonomous positioning capability in any future deep-space probes (similar in mission to Pioneer 10 or 11, or Voyager 1 and 2).

Bibliography

1. Ananda, M. P. et al. "Interplanetary Beacon for Deep Space Navigation," Transactions of the 1980 Position Location and Navigation Symposium. 397-399. Atlantic City, N.J.: IEEE, 1980.
2. Campbell, M. E. "A Celestial Orientation System Based on Star Pattern Recognition," Journal of Spacecraft and Rockets, 2: 962-963 (November-December 1965).
3. Carney, J. K. et al. "Monolithic Optoelectronic/Electronic Circuits," Transactions of the 1982 Gallium Arsenide Integrated Circuit Symposium. 38-40. New Orleans: IEEE, 1982.
4. deCallatay, Vincent. Atlas of the Sky, translated by H. Jones. London: McMillan & Company, 1958.
5. Diederich, Douglas P. and William J. Owen. "An Autonomous System for Satellite Navigation and Attitude Determination," Proceedings of the IEEE Joint Automatic Control Conference. 224-229. New York: IEEE, 1979.
6. Dorman, William J. and Ned E. Clapp, Jr. "A Self-Contained Position Determination System for Low Level Earth Satellites in Nearly Circular Orbits," Proceedings of the Eighth Annual East Coast Conference on Aeronautics and Navigational Electronics. Paper 4.1.2. Baltimore: Institute of Radio Engineers, October 1961.
7. Dzilvelis, Alexander A. "Celestial Map Matching for Space Navigation," Litton Corporation, 1963.
8. Flink, J. H. "Star Identification by Optical Radiation Analysis," Transactions on Aerospace and Navigational Electronics, ANE-10: 212-221. New York: IEEE, September 1963.
9. Instruction Manual Model 280 Video Transceiver. Boulder, Colorado: Colorado Video, Incorporated, November 1978.
10. Instruction Manual Model 720 Programmed Digital I/O Module. Boulder, Colorado: Colorado Video, Incorporated, January 1980.
11. Miller, J. L. and K. C. Daly. "An Approach to the Satellite Navigational Capabilities for the Block 5D Spacecraft," Transactions of the 1980 Position Location and Navigation Symposium. 408-412. New York: IEEE, 1980.

12. Mitton, Simon, editor. The Cambridge Encyclopedia of Astronomy. 457-472. New York: Crown Publishers, 1977.
13. Moskowitz, Saul. "An Approach to Autonomous Orbital Navigation," Transactions on Aerospace and Electronic Systems, AES-4: 736-751. New York: IEEE, September 1968.
14. Nautical Almanac Office, U.S. Naval Observatory. The Air Almanac. Washington, D.C.: US Government Printing Office, 1981.
15. Packard, John N. "Electro-Optical Image Matcher for Space Guidance Applications," Transactions on Aerospace and Navigational Electronics, ANE-10: 282-289. New York: IEEE, September 1963.
16. Pitts, William D., Analysis of Laser Spot Images. MS Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December 1983.
17. Potter, Norman S. "Orientation Sensing in Inertial Space by Celestial Pattern Recognition Techniques," Proceedings of the Fifteenth Annual Meeting of the American Rocket Society. New York: American Rocket Society, December 1960.
18. Randall, Brian. "System Structure for Software Fault-Tolerance," Transactions on Software Engineering, SE-1: 220-232. New York: IEEE, 1975.
19. Tsipis, Kosta. "The Long Range Cruise Missile," Bulletin of the Atomic Scientists. 15-26. (April 1975).
20. White, R. L. et al. "Attitude and Orbit Estimation Using Stars and Landmarks," Transactions in Aerospace and Electronic Systems, AES-11: 195-202. New York: IEEE, March 1975.
21. Whitman, Robert T. "Astroinertial Navigation for Cruise Applications," Hawthorne, California: Northrop Corporation, September 1980. (AD A090 649).
22. Williams, John R. "Hybrid Microcircuit Light Detector Process Technology Development," Proceedings of the 1982 Electronic Components Conference. 105-106. San Diego: IEEE, May 1982.
23. Viglione, S. S. and H. F. Wolf. "Star Field Recognition for Space Vehicle Orientation," Proceedings of the Ninth Annual East Coast Conference on Aerospace and Navigational Electronics. Paper 1.2.5. Baltimore: Institute of Radio Engineers, October 1962.

APPENDIX A:
COMPUTER PROGRAMS DEVELOPED FOR STAR IDENTIFICATION

A1: Pattern Recognition Program: "Whoswho in the heavens".

The primary processing for determining the identity of a star-sample by attempted matches against the templates catalogued in the database (Appendix B). This program uses primarily the geometric pattern analysis of nested recomposition to the polygon-4 level, but also uses a brightness correlation algorithm as a secondary pattern-matching mechanism.

A2: Sample Construction Program: "Frontend sample construction input to pattern recognizer".

Used for phase III and phase IV system validation, this program replaces the manual sample construction process used in phase II validation with an automated processing of a file of digitized brightness values ("pixels").

A3: Video Transceiver Driver: "Video".

Utility used to transfer digitized brightness values from the Colorado Video Transceiver I/O board to the Cromemco Z-2D computer.

```

program whoswhointheheavens(starbase,output);

type indices                      = record
    tempnum,
    hypo,
    suptsamp,
    supttemp : integer;
end;

var
    starbase                : file of char;

    currenttemplate,
    sample                   : array [0..9,1..9] of real;
    points                   : array [0..58] of real;
    magnitude                : array [0..9] of real;
    stalreadypaid            : array [1..9] of boolean;
    geomkey                   : array [1..9] of indices;

    hypoerror, supporterror,
    testerror, m1dist, m2dist,
    scalefactor,
    normalizer,
    triadpoints,
    quadpoints, brightpoints,
    accbriteval, mostbritepoints,
    best, secondbest,
    epsilon                   : real;

    numberofentriesincurrenttemplate,
    samplenummer,
    winner,
    templatecounter,
    hypothesis,
    supportsample,
    supporttemplate,
    lasttemplate, lasthypo,
    lastsuptsamp,
    sslused, ss2used,
    stlused, st2used,
    numberofstarsinsample,
    compatiblecandidates,
    divisor,
    i,j                       : integer;

(* ----- *)
(*                               end of global variables                               *)
(* ----- *)

```

```

(*****
*
*   Function Tolerance:   E.P. Schempp   16 Aug 1983.   *
*   Used to determine if a sample distance value matches *
*       a template distance value from the database   *
*       according to some distance rule (currently     *
*       Menkowski-One).                               *
*   Inputs: the two values to be compared, and a      *
*       constant that represents the scaled percentage *
*       that these values may differ (the cut-off value). *
*   Outputs: the maximum acceptable difference between *
*       the input values that would be considered an  *
*       acceptable "match".                           *
*   No global variables are used (unless passed) and  *
*       none are changed. No files are read or changed. *
*   Calls: nothing.                                   *
*   Called by: mainline code and procedure candidate.  *
*
*****
)

```

```

function tolerance (x,y,epsalon : real) : real;

    var    z      : real;

begin
    z := (x + y) / epsalon;
    if z < 0.5 then z := 0.5;      (* minimum tolerance is *)
    tolerance := z;                (* never < 0.5 degrees. *)
end    (* of function *) ;

(* ..... *)

```

```

(*****
*
* Function Brightnesscorrelation: E.P. Schempp
*                               26 Aug 1983.
* Used to award points to a template that has been
*   matched to the input sample at the polygon-3
*   level based on the brightness of the bodies used
*   to effect the match.
* Inputs: the total number of support-stars in the
*   template that is currently being processed (for
*   scaling the results) and the number of sides used
*   for the comparison. (Note: this function has
*   only been validated for a three-sided match.
* Outputs: the points awarded to the template.
* Global variables used: array magnitude (containing
*   the proportional brightness values); normalizer;
*   brightpoints; supportsample; supporttemplate;
*   lasttemplate; lasthypo; lastsuptsamp; hypothesis;
*   templatecounter; stalreadypaid.
* No files are read or changed.
* Calls: nothing.
* Called by: procedure candidate.
*****
function brightnesscorrelation(ctsize,sides: integer): real;

    var    weighthypo,
           weighttemp,
           weightordinal      : real;
           i                  : integer;

begin
    weighthypo := 2.00;
    weighttemp := 1.00;
    weightordinal := 0.01;

    brightnesscorrelation := (brightpoints / normalizer) *
        ( (magnitude[hypothesis] * weighthypo) +
          (magnitude[supporttemplate] * weighttemp) +
          ((10 - abs(supportsample - supporttemplate)) *
            weightordinal) );

    if ( (lasttemplate = templatecounter) and
        (lasthypo = hypothesis) and
        ( (lastsuptsamp = supportsample) or
          stalreadypaid[supporttemplate] ) ) then
        brightnesscorrelation := 0 (* disallow double
            dippers: follow-on points always less, so
            disallow the lesser of the point totals. *)
    else begin (* record id of payee *)
        lasttemplate := templatecounter;
        lasthypo := hypothesis;
        lastsuptsamp := supportsample;
        stalreadypaid[supporttemplate] := true;
    end; (* else *)
end; (* of function *)
(* ----- *)

```

```

(*****
*
*   Procedure Nexttemplate:   E.P. Schempp   10 Aug 1983 *
*   Used to read in and pre-process the subsequent *
*   template from the database ("starbase"). *
*   Inputs: the number of the next template expected (as *
*   a check). *
*   Outputs: the template parameters globally, and the *
*   number of support-stars as a passed value. *
*   Global variables used: several: this is a *
*   procedure that is fairly "global" in nature, but *
*   written as a separate procedure for ease of *
*   handling. *
*   Files read: "Starbase" (the template database). *
*   No files are changed. *
*   Calls: nothing. *
*   Called by: mainline code. *
*)
*****

```

```

procedure nexttemplate(counter: integer; var size: integer);

```

```

var   i,j,
      indexnumberread,
      maxread,
      rownumber           : integer;
      mostsig             : real;
      tens,units,
      decimalpt,tenths,
      space,ch,ch2        : char;

```

```

begin
  readln(starbase,ch,ch2);  (* header: star index number *)
  indexnumberread := (10 *
    (ord(ch) - ord('0')) ) + (ord(ch2) - ord('0')) );
  if indexnumberread <> counter then
    writeln('error message for non-match, template number: ',
      counter:2);          (* optional check *)

  for i := 1 to 4 do
    read(starbase,ch);
    rownumber := ord(ch) - ord('0');  (* initial rownumber *)
    if rownumber <= 0 then size := 0;
    for i := 1 to 2 do
      read(starbase,ch);  (* discard the "=" characters *)

  while rownumber >= 0 do begin
    (* will be negative for non-numerals *)
    if rownumber >= 8 then
      maxread := 7
      (* needed due to database construction *)
    else
      maxread := rownumber -1;

```

```

for j := 0 to maxread do begin
  read(starbase,tens,units,decimalpt,tenths,space);
  if tens = ' ' then mostsig := 0
  else mostsig := 10;
  (* assuming always a space or a 1 *)
  currenttemplate[j,rownumber] := mostsig +
    (ord(units) - ord('0')) +
    ( (ord(tenths) - ord('0')) ) / 10 );
  end (* of for loop *) ;
  for j := rownumber to 7 do
    for i := 1 to 5 do
      read(starbase,ch);      (* skip empty entries *)
      read(starbase,ch,space); (* discard "% " *)
      read(starbase,units,decimalpt,tenths);
      (* read star magnitude *)
      magnitude[rownumber] := (ord(units) - ord('0')) +
        ( (ord(tenths) - ord('0')) ) / 10);
      size := rownumber;
      (* hold pre-updated value of rownum *)
      readln(starbase);      (* skip to start of next line *)
      for i := 1 to 4 do
        read(starbase,ch);
        rownumber := ord(ch) - ord('0');
        for i := 1 to 2 do
          read(starbase,ch);      (* skip the "=" part *)
        end (* of while loop *) ;
      for i := 2 to size do
        for j := 1 to i-1 do (* fill in transpose entries *)
          currenttemplate[i,j] := currenttemplate[j,i];
        end;
      normalizer := 0;
      for i := 1 to numberofentriesincurrenttemplate do begin
        magnitude[i] :=
          (0.1 + magnitude[numberofentriesincurrenttemplate]
            - magnitude[i]) * (4.8 - magnitude[i]);
        (* scale is now inverted and magnified,
           with the largest values corresponding
           to the brightest stars *)
        normalizer := normalizer + magnitude[i];
      end;
      for j := size+1 to 10 do
        readln(starbase);      (* already read 1st empty line *)
        (* skip empty end of template *)
      end;
    end; (* of procedure nexttemplate *)
  end;
  (* ..... *)

```

```

(*****
*
* Procedure Readsample:      E.P. Schempp      10 Aug 1983 *
* Used to read in and pre-process the sample. This *
* version is interactive. *
* Inputs:  the number of support-stars and the *
*           corresponding distances (sample features) *
*           supplied by the user after the system prompts. *
* Outputs: sets the global variables below: *
* Global variables used: none except array "sample" *
*           and "numberofstarsinsample", which are set. *
* No files are read or changed. *
* Calls: nothing. *
* Called by: mainline code. *
*****)

procedure readsample;

    var i,j      : integer;

begin
    writeln('Enter sample number. ');
    readln(samplenum);
    writeln('Enter the number of stars in this sample. ');
    readln(numberofstarsinsample);
    for i := 1 to numberofstarsinsample do begin
        writeln('Enter the', i:2,
            ' value(s) for the next sample star. ');
        for j := 0 to i-1 do
            read(sample[j,i]);
        readln;
    end;

    for i := 2 to numberofstarsinsample do
        for j := 1 to i-1 do
            sample[i,j] := sample[j,i];      (* transposition *)
        end;
    end;

end      (* of procedure *);

(* ..... *)

```

```

(*****
*
* Procedure Distancerule: E.P. Schempp 22 Aug 1983 *
* Used to compute the desired sample-template *
* distance. Currently this is Menkowski-One and *
* Menkowski-Two at the polygon-3 level. *
* Inputs: the sample-template differences for the *
* individual sides (single measurements). *
* Outputs: polygon-3 (triangle level) distances. *
* Global variables used: none. *
* No files are read or changed. *
* Calls: nothing. *
* Called by: procedure candidate. *
*****)

procedure distancerule (err1,err2,err3 : real;
                        var sigma1,sigma2 : real);

begin
  sigma1 := err1 + err2 + err3;      (* Menkowski One *)
  sigma2 := sqrt ((err1*err1) + (err2*err2) + (err3*err3));
                                     (* Menkowski Two *)
end;  (* of procedure *)

(* ..... *)

```

```

(*****
*
* Procedure Tellhowgood:      E.P. Schempp   11 Sep 1983 *
* Used to determine the name of the template with the *
* best match to the input sample and to provide an *
* indication of the value of this match. *
* Inputs:  the highest two template point totals and *
* catalog number of the selected template. *
* Outputs: to the screen: the catalog number and name *
* of the selected star, and a high, medium, or low *
* confidence rating. *
* Global variables used: none. *
* Files read: the template database "starbase". *
* No files are changed. *
* Calls: nothing. *
* Called by: mainline code. *
*)

```

```

procedure tellhowgood(first,second: real; topstar: integer);

    var i,j    : integer;
        ch     : array [1..20] of char;
begin
    reset(starbase);
    j := 11 * (topstar-1);
    for i := 1 to j do
        readln(starbase);
    read(starbase,ch[1],ch[2]);
    i := 2;
    while ( (ch[i-1] <> ' ') or (ch[i] <> ' ') ) do begin
        (* find double space *)
        i := i + 1;
        read(starbase,ch[i])
    end;
    writeln;
    writeln('Sample identified as catalog-star number ',
            ch[1], ch[2], ':');
    write('
for j := 5 to i do
    write(ch[j]);
writeln; writeln;

if ( (first > 200) or ((first/second) > 3.3) or
    ( (first > 135) and ((first/second) > 2.0) ) ) then
    writeln('Confidence level is:    HIGH.')
else
    if ( (first > 150) or ( (first/second) > 1.7) ) then
        writeln('Confidence level is: Moderate.')
    else
        writeln('Confidence level is:    low. ');
end; (* of procedure *)
(* ..... *)

```

```

(*****
*
* Procedure Candidate:      E.P. Schempp      19 Jul 1983 *
* Controls all processing after a successful sample- *
* template match at the polygon-3 level. *
* This is effectively a "global" area of code that is *
* written as a procedure to facilitate handling and *
* visualization of the overall program. All major *
* variables are therefore global with the mainline *
* code, and there are no strict inputs or outputs. *
* No files are read or changed. *
* Calls: brightnesscorrelator. *
* Called by: mainline code. *
*
*****
)

procedure candidate;

  var  i,j  : integer;

begin
  testerror := abs (sample[1,supportsample] -
                    currenttemplate[hypothesis,supporttemplate]);
  distancerule(hypoerror, supporterror, testerror,
               m1dist, m2dist);
  scalefactor := (sample[0,1] + sample[0,supportsample] +
                  sample[1,supportsample] +
                  currenttemplate[0,hypothesis] +
                  currenttemplate[0,supporttemplate] +
                  currenttemplate[hypothesis,supporttemplate] ) / 6;
  (* scale the error values: *)
  hypoerror := (2 * hypoerror) /
               (sample[0,1] + currenttemplate[0,hypothesis]);
  supporterror := (2 * supporterror) /
                 (sample[0,supportsample] +
                  currenttemplate[0,supporttemplate]);
  testerror := (2 * testerror) / (sample[1,supportsample] +
                                   currenttemplate[hypothesis,supporttemplate]);
  distancerule(hypoerror, supporterror, testerror,
               m1dist, m2dist);

  (* optional display of scaled polygon-3 level distances:
  writeln('scaled errors - hypo/supt/test:', hypoerror:8,
          '/', supporterror:8, '/', testerror:8);
  writeln('scaled totalerrors - m1 .. m2: ', m1dist:9,
          '..', m2dist:9);
  writeln('CT size/scalefactor: ',
          numberofentriesincurrenttemplate,
          '/', scalefactor:11);
  writeln('POINTS now =', points[templatecounter]:13);
  *)

```

```

if ( (geomkey[compatiblecandidates].tempnum =
      templatecounter) and
      (geomkey[compatiblecandidates].hypo = hypothesis) )
then
  compatiblecandidates := compatiblecandidates + 1
else begin
  (* process and empty stack *)
  for i := 1 to compatiblecandidates do (* begin *)
    for j := i+1 to compatiblecandidates do (* begin *)
      if ((geomkey[i].suptsamp <> geomkey[j].suptsamp)
          and
          (geomkey[i].supttemp <> geomkey[j].supttemp)
          and
          ( ( (st1used <> geomkey[i].supttemp) or
              (st2used <> geomkey[j].supttemp) ) and
            ( (st1used <> geomkey[j].supttemp) or
              (st2used <> geomkey[i].supttemp) ) and
            ( (ss1used <> geomkey[i].suptsamp) or
              (ss2used <> geomkey[j].suptsamp) ) and
            ( (ss1used <> geomkey[j].suptsamp) or
              (ss2used <> geomkey[i].suptsamp) ) )
            (* these disallow the reuse of an
              already successful sample-sample or
              template-template confirmatory pair
              within the same hypothesis-support-
              star. This is a single-time memory
              capability. *)
            and
            ( (abs(sample[geomkey[i].suptsamp,
              geomkey[j].suptsamp] -
              currenttemplate[geomkey[i].supttemp,
              geomkey[j].supttemp]) )
              <= tolerance(sample[geomkey[i].suptsamp,
              geomkey[j].suptsamp],
              currenttemplate[geomkey[i].supttemp,
              geomkey[j].supttemp], epsilon) ) )
            then begin
              points[templatecounter] :=
                points[templatecounter] + quadpoints;
              (* record the pairings used: *)
              ss1used := geomkey[i].suptsamp;
              ss2used := geomkey[j].suptsamp;
              st1used := geomkey[i].supttemp;
              st2used := geomkey[j].supttemp;

              (* optional diagnostic:
                writeln;
                writeln(' QuadMatch:  star no. ',
                  templatecounter:3, ' points now = ',
                  points[templatecounter]:10); *)
            end;
          (* if. no else *)
        (* end of j for-loop *)
      (* end of i for-loop *)
      compatiblecandidates := 1;      (* re-initialize *)
    end;
  (* else. also endif *)

```

```

                (* always: *)
geomkey[compatiblecandidates].tempnum := templatecounter;
geomkey[compatiblecandidates].hypo    := hypothesis;
geomkey[compatiblecandidates].suptsamp := supportsample;
geomkey[compatiblecandidates].supttemp := supporttemplate;
(* some redundancy in here, since the last candidate data
   is stored even after a process-and-dump, for comparison
   use on the next candidate entry, which will always be
   false and therefor discarded. *)

end; (* of procedure *)

```

```

(* ===== *)
(*                                     *)
(*               End of Procedures   *)
(*                                     *)
(* ===== *)
(*                                     *)
(*               Start of Main Code  *)
(*                                     *)
(* ===== *)

```

```

(* ===== *) begin (* ===== *)

reset(starbase);
numberofentriesincurrenttemplate := 0;
secondbest := 1;
supportsample := 1; (* band-aid *)
supporttemplate := 1;
compatiblecandidates := 1;
epsilon := 21.3; (* set measurement tolerance to 9.4
                  per cent : tolerance = 2/epsilon. *)
triadpoints := 0.0; (* three-sided match weighting factor:
                    set zero since star gets some bright-
                    points for every successful triad *)
quadpoints := 100.0; (* four-side geometry weighting *)
brightpoints := 30.0; (* brightness correlation weight *)

readsample;
writeln;
writeln(' SAMPLE NUMBER: ', samplenum);
writeln(' ===== ');

```

```

for templatecounter := 1 to 57 do begin
  accbriteval := 0; (* initialize *)
  mostbritepoints := 0;
  nexttemplate (templatecounter,
                numberofentriesincurrenttemplate);
  for hypothesis := 1 to numberofentriesincurrenttemplate
    do begin
    if abs (sample[0,1] - currenttemplate[0,hypothesis])
      <= tolerance(sample[0,1],
                  currenttemplate[0,hypothesis], epsilon)
    then begin
      for supportsample := 2 to numberofstarsinsample do
        (* begin *)
        for supporttemplate := 1 to
          numberofentriesincurrenttemplate do
          if ( (supporttemplate <> hypothesis) and
              (abs (sample[0,supportsample] -
                  currenttemplate[0,supporttemplate])
              <= tolerance (sample[0,supportsample],
                          currenttemplate[0,supporttemplate],
                          epsilon) ) )
          then begin
            hypoerror := abs(sample[0,1] -
                            currenttemplate[0,hypothesis]);
            (* moved inside this loop to counter
              scaling in candidate *)
            supporterror := abs(sample[0,supportsample]
                              - currenttemplate[0,supporttemplate]);
            if abs (sample[1,supportsample] -
                  currenttemplate[hypothesis,
                                supporttemplate] )
              <= tolerance (sample[1,supportsample],
                          currenttemplate[hypothesis,
                                supporttemplate], epsilon)
            then begin
              points[templatecounter] :=
                points[templatecounter] + triadpoints;
              accbriteval := accbriteval +
                brightnesscorrelation
                  (numberofentriesincurrenttemplate,3);
              candidate;
            end (* if. no else *)
          end (* if. no else *)
        (* end; supporttemplate for-loop: 1 stmt. *)
      (* end; supportsample for-loop: 1 stmt. *)
    end;
    (* if. no else *)
  end;

```

```

for j := 1 to 9 do
    stalreadypaid[j] := false;
    sslused := 0;
    ss2used := 0;
    stlused := 0;
    st2used := 0;
    (* optional diagnostic on award of brightness points:
    if accbriteval > 0 then
        writeln('STAR-template', templatecounter:3,
            ' with support-star ', hypothesis:2,
            ' brite-points totaled', accbriteval:13, '.'); *)

    if accbriteval > mostbritepoints then
        mostbritepoints := accbriteval;
        accbriteval := 0;
    end;
    candidate; (* hypothesis for-loop *)
    (* before next template to empty
    any current stack *)
    (* optional diagnostic on award of geometric points:
    if points[templatecounter] > 0 then
        writeln('i/geometrypnts:', templatecounter:3, ' / ',
            points[templatecounter]:13); *)
    if numberofentriesincurrenttemplate <=
        numberofstarsinsample then
        divisor := numberofentriesincurrenttemplate
    else
        divisor := numberofstarsinsample;
        (* divide by minimum of these 2 *)
    points[templatecounter] :=
    points[templatecounter] / divisor; (* scaled results *)
    points[templatecounter] :=
    points[templatecounter] + mostbritepoints;

    if (points[templatecounter] > best) then begin
        secondbest := best;
        best := points[templatecounter];
        winner := templatecounter;
    end
    else
        if (points[templatecounter] > secondbest) then
            secondbest := points[templatecounter];

end; (* of templatecounter for-loop *)

writeln;
tellhowgood(best,secondbest,winner);

(* ===== *) end. (* ===== *)

```

```

(*****
*
* Frontendsampleconstructioninputtopatternrecognizer
*   Program           E.P. Schempp           29 Sep 1983
* This program is used to create a sample, suitable for
* input directly to the main pattern recognition
* program, from a data file of brightness values.
* It first prompts the user for a threshold brightness
* value ('baselevel') representative of the level of
* background light detected by the sensor: only
* brightness values above this threshold level are
* processed as star segments. The baselevel should be
* entered as a single ASCII character followed by a
* carriage return. If the level entered is too low to
* allow the construction of a reasonable number of
* stars, 'baselevel' is decremented and processing
* starts over. A second user prompt requests a value
* for scaling the derived distances from the original
* image. The default for this is the standard value
* used for the Dage 650 TV camera (10.4).
* Language: Pascal.
* Operating System: UC Berkeley Unix version 4.1.
* System: AFIT SSC (Vax 11/780).
*
*****)

```

```

program frontendsampleconstructioninputtopatternrecognizer
      (pixels,output);

```

```

type pointdata      = array [0..255] of record
                        ycoord, xcoord,
                        bright      : integer;
                        end;

```

```

var  pixels          : file of char;

     sample          : array [0..9,0..9] of real;

     britespot,
     star            : pointdata;

     scalefactor     : real;

     i,j,
     starcount,
     counter         : integer;

     point, baselevel : char;

```

```

(* ----- *)
(*           end of global declarations           *)
(* ----- *)

```

```

(*****
*
* Procedure Getpoints:      E.P. Schempp      23 Sep 1983  *
* Used to test the entire file of raw data "pixels" on a  *
* point-by-point basis against a threshold level. In      *
* this version the program prompts the user to input a    *
* value ("baselevel") since this allows                   *
* experimentation with the television camera and video    *
* transceiver adjustments for different conditions and    *
* testing phases. This level is always decremented as    *
* the processing is completed, so that the mainline      *
* code can repeatedly recall it if the number of         *
* points found to be above the threshold is too few      *
* for adequate processing.                                *
* Inputs:  the brightness threshold supplied by the user  *
* after prompting.                                         *
* Outputs: the array "britespots" filled in with the x    *
* and y coordinates and the brightness values of all      *
* points in "pixels" that are above threshold. Also a    *
* count if the number of these points ("counter").        *
* Global variables used: counter, britespot, baselevel.  *
* Files Read: pixels. No files are changed.              *
* Calls: nothing.                                          *
* Called by: mainline code.                               *
*
*****)
procedure getpoints;

      var    x,y      : integer;

begin

  writeln('baselevel is ', baselevel);
  reset(pixels);
  counter := 0;
  for y := 0 to 127 do      (* begin *)
    for x := 0 to 255 do    begin
      if not eof(pixels) then
        read(pixels,point);
      if (point > baselevel) then begin
        britespot[counter].xcoord := x;
        britespot[counter].ycoord := y;
        britespot[counter].bright := ord(point) -
                                     ord(baselevel);

        counter := counter + 1;
      end;      (* if. no else *)
    end;      (* x for-loop *)
  (* end; y for-loop : single statement *)
  britespot[counter].ycoord := 9999;
  (* ensures a final dump in joinspots *)
  baselevel := chr(ord(baselevel) -1);
  (* decrement baselevel *)

end; (* of getpoints *)

```

```

(*****
*
* Procedure Joinspots:      E.P. Schempp      29 Sep 1983
* Used to join the separate points in the array
* "britespot" into the correct multi-point bright
* bodies. It determines which of the points belong
* to the same star, along with the brightest single
* element in a star, and "collapses" the star into a
* single point that contains the sum of all of the
* brightness values for this star. The procedure
* places the coordinate pairs on a "heap" until it
* finds a jump in the x coordinate: this signifies a
* vertical gap between the bright points that means
* star(s) placed on the heap so far have ended. The
* additional possibility that a point with a close
* x coordinate but a different y coordinate,
* indicating an additional star that may be either
* after or in the midst of the a star (or group of
* stars) presently on the heap, is also checked. When
* detected, the "intruding" star is placed on a second
* heap ("skipped") for later processing by a recursive
* call to joinspots.
* Inputs: the array "brightspots" created by getpoints
* and the counter that denotes its size.
* Outputs: the array "britespots" reduced to the number
* and location of actual bodies that were in the
* original image.
* Global variables used: starcount.
* No files are read or changed.
* Calls: joinspots (recursive).
* Called by: mainline code.
*
*****)

```

```

procedure joinspots (bs : pointdata; counter : integer);

    var i,j,k,m,last,max,
        skippedcount,
        topofheap,
        bottomofheap      : integer;
        heap               : array [0..30] of integer;
        skipped            : pointdata;
        heapmatch          : boolean;

begin

    skippedcount := 0;
    heap[0] := 0;
    topofheap := 0;    bottomofheap := 0;

    for i := 1 to counter do begin
        last := i - 1;
        if ( (bs[i].ycoord - bs[last].ycoord) >= 2 ) then begin
            (* found a star-gap: do heap *)
            starcount := starcount + 1;

```

```

max := toposheap - bottomofheap;
for j:= ((topofheap - bottomofheap) -1) downto 0 do
  if (bs[heap[j]].bright > bs[heap[max]].bright)
    then max := j;
    (* endif. no else. end; j for-loop *)
  for k := (topofheap - bottomofheap) downto 0 do
    star[starcount].bright:= star[starcount].bright
      + bs[heap[k]].bright;
  star[starcount].ycoord := bs[heap[max]].ycoord;
  star[starcount].xcoord := bs[heap[max]].xcoord;
  writeln('newstar:', starcount:3, ' at ',
    star[starcount].ycoord:4,
    star[starcount].xcoord:4,
    ' =', star[starcount].bright:3);
  toposheap := i;
  bottomofheap := i;
  heap[0] := i;      (* start next heap *)
  skipped[skippedcount].ycoord := 9999;
                        (* skipped stopper *)
  joinspots(skipped, skippedcount); (* recursive *)
  skippedcount := 0;
end
else begin      (* no gap detected *)
  heapmatch := false;
  (* start test for neighbor vs. intruder *)
  for m:= (topofheap - bottomofheap) downto 0 do
    if ( (abs(bs[i].xcoord - bs[heap[m]].xcoord))
      <= 1) then (* using the = includes
        diagonals. *)
      heapmatch := true;
      (* no else. single statement for-loop *)
    if ( ((bs[i].ycoord =
      bs[heap[topofheap-bottomofheap]].ycoord) and
      ((bs[i].xcoord -
      bs[heap[topofheap-bottomofheap]].xcoord) = 1))
      or
      (((bs[i].ycoord -
      bs[heap[topofheap-bottomofheap]].ycoord) = 1)
      and heapmatch) ) then begin
        toposheap := toposheap +1;
        heap[topofheap - bottomofheap] := i;
      end
    else begin
      (* close ycoord but invalid x match *)
      skipped[skippedcount].bright:= bs[i].bright;
      skipped[skippedcount].ycoord:= bs[i].ycoord;
      skipped[skippedcount].xcoord:= bs[i].xcoord;
      skippedcount := skippedcount +1;
    end; (* else *)
  end; (* else. endif *)
end; (* i for-loop *)

end; (* of joinspots *)

```

```

(*****
*
* Procedure Makesample:      E.P. Schempp      27 Sep 1983
* Used to compute the Euclidean distance between the star
* midpoints resolved by procedure joinspots.
* Inputs:  the array "star" and the counter "starcount".
* Outputs: the array "sample".
* Global variables used:  the inputs and outputs above.
* No files are read or changed.
* Calls: nothing.
* Called by:  mainline code.
*
*****)

```

```

procedure makesample;

```

```

    var  i,j,k,trystar      : integer;
         index              : array [0..25] of integer;

```

```

begin

```

```

    for i := 0 to 25 do begin (* 25 stars is enough *)
        trystar := 1;
        for j := 2 to starcount do
            if star[j].bright > star[trystar].bright then
                trystar := j;
        index[i] := trystar;
        star[trystar].bright := -1 * star[trystar].bright;
        (* preserve magnitude but remove from further use *)
    end; (* i for-loop *)
    for i := 0 to 25 do
        write(index[i]:3); (* optional: output list of
                           joined stars *)

```

```

    writeln;

```

```

    for i := 0 to 9 do begin
        for k := 0 to (i - 1) do (* compute Euclidean dist. *)
            sample[k,i] := sqrt( ((star[index[i]].xcoord -
                                   star[index[k]].xcoord) *
                                   (star[index[i]].xcoord -
                                   star[index[k]].xcoord)) +
                                   ((star[index[i]].ycoord -
                                   star[index[k]].ycoord) *
                                   (star[index[i]].ycoord -
                                   star[index[k]].ycoord)) )

```

```

                                   / scalefactor;

```

```

        (* A squared plus B squared equals C squared *)

```

```

        if sample[0,i] > 10.0 then

```

```

            i := i - 1;

```

```

            (* ignore any stars outside of ten degrees *)

```

```

        end; (* of i for-loop *)

```

```

end; (* of makesample *)

```

```

(* ===== *)
(*                                     *)
(*               End of Procedures   *)
(*                                     *)
(* ++++++ *)
(*                                     *)
(*               Start of Mainline Code
(*                                     *)
(* ===== *)

```

```

(* ===== *) begin (* ===== *)

writeln('Enter baselevel (character). ');
readln(baselevel);
writeln('Enter sensor scalefactor (real number). ');
scalefactor := 10.4;
readln(scalefactor);

while counter < 12 do
    getpoints;
    for i := 0 to counter do
        writeln(i, ': ', britespot[i].ycoord:4,
            britespot[i].xcoord:4, '=', britespot[i].bright:3);
    writeln;
    joinspots(britespot, counter);
    writeln;
    writeln;
    makesample;
    for i := 1 to 9 do begin
        for j := 0 to (i-1) do
            write(sample[j,i]:11);
            writeln;
        end;
    end;

(* ===== *) end. (* ===== *)

```

```

/*****
*
* Program Video.C          E.P. Schempp          28 Sep 1983
* This program is used to strobe the pixel data
* from the Colorado Video Corporation Model 280 Video
* Transceiver and dump it onto a diskette through the
* AFIT School of Engineering Signal Processing Lab
* Cromemco Z-2D System Two computer system. The sequence
* of processing follows the documentation for the Video
* Transceiver and the Model 280 I/O Module, as well as
* the Cromemco Tu-Art Digital Interface manual:
* the program proceeds across a horizontal line (varying
* x addresses) until all 256 pixels are transferred, then
* proceeds to the next lower horizontal line (increments
* the y address) and repeats until finished. After each
* four horizontal lines are transferred to the Cromemco
* internal memory, the 1024 values are written to a disk
* file. For each of the 64K addresses, the computer sends
* a control sequence out the Tu-Art parallel port address
* 064 hex, then follows this with the desired x address
* of the current pixel data out parallel port address 094
* hex. A signal from the Transceiver then temporarily
* halts the computer. When allowed to continue, a
* sequence and value for the y address is similarly
* output to the parallel ports. The computer is again
* halted by the Transceiver while it outputs the pixel
* value back to the computer ("z data"). This value is
* read in the 064 port, and retained in array "pixels"
* until transferred to disk. The process continues until
* 64K (or less, if modified to omit picture border areas
* or the bottom half of the data field, for example) have
* been transferred.
* Language: BDS "C" version 1.4.
* Operating System: Cromemco Disk Operating System
* ("CDOS") version 02.36
*
*****/

```

```

/* global declarations: */

#include "bdscio.h" /* standard C header file */

int fd; /* file directory mnemonic */
char pixels[1024]; /* main data structure for the
received brightness values */

```

```

/***** begin program *****/

main()

{
    int    i,j,k,x,y,msb,delay;
    int    j3;
    int    raw;

    printf("Program start.\n\n");
    fd = creat("D:PIXELS.H");          /* open a disk file */

    for (i=1; i<32; i++)                /* 0/64 for whole picture */
    {                                    /* main loop, 1K at a time */
        printf(" i= %d\n", i);          /* program progress output */
        for (j=0; j<4; j++)             /* do 4 horiz. lines per dump */
        {
            y = (2 * ((4*i) + j) );    /* doubling the y addresses */
            /* printf("      j/y= %d/%d\n", j, y); */
            for (x=0; x<256; x++)        /* do all values on a
                                           horizontal line */
            {
                outp(0x94,0x30);        /* control bits: 0011 XXXX */
                j3 = (0x00FF & x);        /* mask extraneous bits */
                outp(0x64,j3);          /* send control to Colo. */
                for (delay=0; delay<5; delay++)
                    ;                    /* do nothing until the wait.

                Finish the count when re-awakened, then continue: */
                if (y < 256)
                {   msb = 0x40;          /* 0100 XXXX */
                    else
                        msb = 0xC0;        /* 1100 XXXX: msb for y */
                outp(0x94,msb);          /* control bits for y address */
                outp(0x64,y);            /* send control to Colo. */
                for (delay=0; delay<16; delay++)
                    ;                    /* do nothing until the wait:

                When re-awakened, continue: */
                raw = inp(0x64);          /* read raw z data */
                pixels[x + (256*j)] = (raw/2) + 64; /* printable*/
            } /* end x for-loop */
        } /* end j for-loop */
        printf(" Start diskdump; x/y = %d/%d.\n", x,y);
        write(fd,pixels,8);             /* 8 128-byte blocks to disk */
    } /* end i for-loop */

    close(fd); /* close the disk file */
    printf("\n\n..... FINISHED .....");
    exit(0);

}

```

Appendix B: Data Base of Catalog-Star Templates

01. Alpharatz - 2.2 / andromeda alpha

1=	6.9						%	3.5	(delta)
2=	9.8	9.1					%	4.4	(theta)
3=	7.7	3.0	6.4				%	4.4	(pi)
4=	7.9	7.3	2.0	5.0			%	4.5	(sigma)
5=	10.0	6.9	15.8	9.8	14.0		%*	4.5	(zeta)
6=	4.7	10.9	14.0	12.1	12.5	11.3	%*	4.6	(pegasus psi) + 4.5

02. Ankaa - 2.4 / phoenix alpha

1=	8.3						%	3.4	(beta)
2=	1.4	7.8					%	3.9	(chi)
3=	4.6	9.9	3.7				%	3.9	(epsilon)
4=	9.4	16.7	9.3	7.0			%*	4.5	(iota)
5=	4.7	4.3	3.7	5.6	12.3		%*	4.5	(mu)
6=	9.3	15.3	8.8	5.6	3.0	11.2	%*	4.6	(theta)
7=	9.4	2.0	8.8	11.6	18.2	5.9 17.1	%*	4.6	(nu)

03. Schedar - 2.1 to 2.6 / cassiopia alpha

ODMBAO

1=	5.0						%	1.6-2.3	(beta)
2=	4.7	6.4					%	2.4	(gamma)
3=	7.1	9.8	3.8				%	3.0-3.1	(delta)
4=	1.8	5.6	3.0	5.4			%	3.6	(eta)
5=	2.6	6.6	7.1	9.2	4.3		%	3.7	(zeta)
6=	6.3	4.8	3.7	6.9	5.4	8.7	%	4.2	(chi)
7=	4.7	9.4	5.7	5.4	4.1	5.2 8.9	%	4.5	(theta)
8=	6.5	2.7	8.6	12.3	7.6	7.1 7.3 11.1	%	4.6	: 4.4-5.1 (rho)

04. Diphda - 2.2 / cetus beta

1=	10.0						%	3.6	(eta)
2=	9.6	17.3					%*	4.6	(2)
3=	6.9	15.8	3.1				%*	4.6	(7)

05. Achernar - 0.6 / eridanus alpha

1=	5.2						%	3.0	(hydrus alpha)
2=	6.0	10.1					%	3.7	(chi)
3=	7.6	10.5	3.0				%	3.8	(phi)
4=	8.0	12.8	4.6	7.3			%	4.0	(phoenix delta)
5=	4.3	8.9	7.5	10.2	6.7		%	4.1	(phoenix zeta)
6=	6.8	10.2	11.3	13.8	10.4	4.0	%	4.5	(phoenix eta)

06. Hamal - 2.2 / aries alpha

1=	3.8						%	2.7	(beta)
2=	7.1	9.0					%	3.6	(triangulum alpha)
3=	9.5	13.0	11.7				%	3.7	(41)

07. Acamar - 3.1 / eridanus theta

1= 3.4
2= 4.9 7.9
3= 9.3 8.5 10.1
4= 8.2 7.7 12.2 16.0

% 4.1 (iota)
% 4.3 (epsilon)
% 4.4 (chi)
% 4.5 (fornax beta)

08. Menkar - 2.8 / cetus alpha

1= 4.6
2= 8.4 12.3
3= 7.4 11.4 1.0
4= 6.5 3.1 14.6 13.7
5= 9.2 6.4 14.0 13.4 8.6
6= 7.3 6.8 9.9 9.5 9.7 4.4
7= 9.1 13.1 9.7 9.1 13.7 18.2 15.7

% 3.6 (delta)
% 3.8 (taurus xi)
% 3.8 (taurus omicron)
% 4.0 (delta)
% 4.3 (xi-2)
% 4.4 (mu)
% 4.4 (taurus 10)

09. Mirfak - 1.9 / perseus alpha

1= 9.6
2= 4.9 12.8
3= 3.8 9.3 8.4
4= 8.0 15.3 3.2 11.4
5= 8.3 7.1 12.9 5.2 16.0
6= 7.7 12.8 11.7 4.3 14.6 6.7
7= 5.8 4.1 8.8 6.6 11.6 6.9 10.7

% 2.2-3.5 (beta)
% 3.1 (gamma)
% 3.1 (delta)
% 3.9 (eta)
% 3.9 (nu)
% 4.0 (48)
% 4.0 (chi) ((++))

10. Aldebaran - 1.0-1.1 / taurus alpha

1= 2.0
2= 9.5 7.7
3= 3.3 3.4 9.5
4= 4.1 2.2 5.7 4.2
5= 3.3 2.2 7.3 2.1 2.1
6= 2.9 2.1 8.2 1.5 2.8 0.8
7= 6.7 7.7 14.4 5.0 9.0 7.1 6.4
8= 4.0 4.1 9.1 7.1 5.5 6.3 6.3
9= 5.9 7.1 13.6 8.8 9.1 9.0 8.8

% 3.6+4.0 (theta2 + 1)
% 3.5-4.0 (lambda)
% 3.6 (epsilon)
% 3.9 (gamma)
% 3.9 (delta)
% 4.2 (68)
% 4.3 (tau)
% 4.3 (90)
% 4.3 (orion ?)

11. Rigel - 0.3 / orion beta

1= 8.9
2= 9.0 1.3
3= 8.0 9.2 8.2
4= 8.8 1.5 3.0 10.2
5= 5.5 4.8 4.4 4.8 5.7
6= 3.6 7.8 8.4 10.6 7.6 6.6
7= 8.0 16.1 15.9 10.5 16.0 11.5 11.1

% 1.8 (epsilon)
% 2.1+4.2 (zeta)
% 2.2 (chi)
% 2.5+2.6 (delta)
% 2.9 (iota)
% 2.9 (eridanus beta)
% 3.3 (lepus mu)

12. Capella - 0.2 / auriga alpha

1=	7.4						% 2.1-2.2 (beta)
2=	5.3	10.1					% 3.3 (eta)
3=	3.5	10.0	2.8				% 3.3-4.1 (epsilon)
4=	9.0	6.1	8.5	10.1			% 4.2 (nu)
5=	8.2	13.0	12.5	9.9	16.8		% 4.4 (camelopardus 7)
6=	8.6	15.2	5.6	5.4	14.2	13.0	% 4.5 (perseus 58)

13. Bellatrix - 1.7 / orion gamma

1=	6.9						% 0.4-1.3 (alpha)
2=	7.4	9.2					% 1.8 (epsilon)
3=	8.5	9.3	1.3				% 2.1+4.2 (zeta)
4=	6.5	8.9	1.5	3.0			% 2.5+2.6 (delta)
5=	8.5	14.7	13.8	15.1	12.5		% 3.3 (pi-3)
6=	7.9	12.8	2.9	3.7	2.9	11.2	% 3.4 (eta)
7=	7.7	14.6	10.0	11.0	9.0	4.4	% 3.6-3.7 (pi-5)
					8.0		(+(3.7), 2(3.8), <11(4.0-4.5))

14. Elnath - 1.8 / taurus beta

1=	7.5						% 2.9 (auriga iota)
2=	7.5	14.8					% 3.0 (zeta)
3=	8.8	15.8	5.9				% 4.3 (gemini 1)
4=	5.3	12.1	7.0	4.5			% 4.5 (136)

15. Alnilam - 1.8 / orion epsilon

1=	8.4						% 0.3 (beta)
2=	9.7	18.1					% 0.4-1.3 (alpha)
3=	7.6	14.1	7.0				% 1.7 (gamma)
4=	1.3	8.6	9.9	8.8			% 2.1+4.2 (zeta)
5=	8.9	7.7	16.9	16.4	8.0		% 2.2 (chi)
6=	1.5	8.4	9.8	6.8	3.0	9.9	% 2.5+2.6 (delta)
7=	4.8	5.3	13.7	12.2	4.3	4.8	% 2.9 (iota)
8=	7.7	3.4	16.5	11.6	8.2	10.3	% 2.9 (eridanus beta)

16. Betelgeuse - 0.4 to 1.2 / orion alpha

1=	7.0						% 1.7 (gamma)
2=	9.6	7.7					% 1.8 (epsilon)
3=	9.9	8.6	1.3				% 2.1+4.2 (zeta)
4=	9.3	6.5	1.5	3.0			% 2.5+2.6 (delta)
5=	5.3	4.3	10.9	11.6	10.2		% 3.7 (lambda)
6=	2.8	9.1	12.2	12.5	11.9	6.0	% 4.2 (mu)
7=	5.9	1.4	7.0	8.0	6.2	4.1	% 4.3 (32)

17. Canopus - -0.9 / carina alpha

1=	4.5						% 2.8 (puppis tau)
2=	9.8	7.6					% 3.2 (puppis nu)
3=	9.6	10.9	18.6				% 3.3 (pictor alpha)
4=	5.9	9.5	11.6	13.3			% 3.9 (pictor beta)
5=	6.2	10.4	15.0	9.1	5.0		% 4.4 (pictor gamma)

18. Sirius - -1.6 / canis major alpha

1=	6.5						% 2.0 (beta)
2=	8.2	10.7					% 3.1 (omicron-2)
3=	4.1	9.7	8.3				% 4.1 (gamma)
4=	3.5	3.4	7.5	7.3			% 4.1 (nu-2)
5=	7.8	9.1	2.1	8.8	6.2		% 4.1 (omicron-1)
6=	5.1	9.3	11.8	4.3	8.3	12.1	% 4.3 (theta)
7=	7.6	5.9	6.9	10.7	4.5	4.8	% 4.4 (script E -1)
8=	2.3	7.5	6.9	2.4	4.9	7.1	% 4.4 (iota)
							% (+4.5)

19. Adhara - 1.6 / canis major epsilon

1=	3.3						% 2.0 (delta)
2=	5.3	4.3					% 2.4 (eta)
3=	8.8	10.7	7.7				% 2.7 (puppis pi)
4=	8.0	10.6	13.3	13.0			% 3.1 (zeta)
5=	5.2	3.0	7.2	13.4	10.9		% 3.1 (omicron-2)
6=	4.3	1.5	3.2	10.1	11.8	4.1	% 3.8 (omega)
7=	4.1	7.3	7.9	7.0	6.4	9.2	% 3.8 (chi)
						8.0	

20. Procyon - 0.5 / canis minor alpha

1=	4.4						% 3.1 (beta)
2=	10.0	12.2					% 3.8 (cancer beta)
3=	9.0	9.6	18.5				% 4.1 (monoceros delta)

21. Pollux - 1.2 / gemini beta

1=	4.7						% 2.0+2.9 (alpha)
2=	8.5	10.5					% 3.5 (delta)
3=	3.7	7.9	6.2				% 3.7 (chi)
4=	4.4	4.7	6.1	5.4			% 3.9 (iota)
5=	5.2	1.2	10.2	8.2	4.2		% 4.2 (rho)
6=	2.5	5.1	6.2	3.1	2.5	5.2	% 4.2 (upsilon)
7=	1.0	3.7	8.7	4.7	4.0	4.2	% 4.3 (sigma)
						2.6	

22. Avior - 1.7 / carina epsilon

1= 5.8
 2= 7.3 6.5
 3= 9.7 5.9 4.4
 4= 7.6 7.8 13.3 13.4
 5= 6.7 5.8 0.9 4.4 12.5
 6= 9.8 7.3 3.0 2.5 14.8 3.6
 7= 7.1 2.1 8.3 6.9 7.0 7.6 8.8
 8= 6.8 11.6 9.1 13.1 13.7 8.9 12.1

% 2.0 (vela delta)
 % 2.3 (iota)
 % 2.6 (vela chi)
 % 3.6 (chi)
 % 3.6 (a)
 % 3.4-4.2 (vela N)
 % 3.7 (vela omicron)
 % 3.7 (volans beta)

23. Suhail - 2.2 / vela lambda

1= 5.2
 2= 9.7 11.2
 3= 4.7 9.7 10.7
 4= 4.5 8.7 7.4 3.4
 5= 3.8 8.2 12.5 3.2 5.7

% 3.6 (psi)
 % 4.0 (pyxis beta)
 % 4.1 (a)
 % 4.1 (d)
 %*4.4 :list. 3.7 (c)

24. Miaplacidus - 1.8 / carina beta

1= 5.8
 2= 5.4 5.7
 3= 5.8 8.6 11.0
 4= 7.9 13.3 12.3 7.6
 5= 9.2 8.5 12.7 6.6 13.9
 6= 8.2 13.6 10.6 10.8 5.0 16.6
 7= 7.1 9.6 4.1 12.9 11.8 15.8 8.3

% 3.2 (upsilon)
 % 3.6 (omega)
 % 3.7 (volans beta)
 % 3.9 (volans zeta)
 % 4.0 (c)
 % 4.1 (chamaeleon alpha)
 % 4.1 : 3.6-4.8 (1)
 % (+2(4.2))

25. Alphard - 2.2 / hydra alpha

1= 8.2
 2= 8.2 14.0
 3= 7.7 2.0 14.3

% 4.1 (iota)
 % 4.3 (upsilon-1)
 % 4.5 (tau-2)

26. Regulus - 1.3 / leo alpha

1= 8.0
 2= 5.0 4.2
 3= 6.0 12.5 8.6
 4= 6.0 10.7 9.4 11.5

% 2.6+3.8 (gamma)
 % 3.6 (eta)
 % 3.8 (omicron)
 % 3.9 (rho)

27. Dubhe - 2.8 / ursa major alpha

1= 5.5
 2= 9.7 10.3
 3= 8.2 13.3 15.0

% 2.4 (beta)
 % 3.9 (upsilon)
 % 4.1 (draco lambda)

28. Denebola - 2.2 / leo beta

1=	8.5					% 3.4 (theta)
2=	7.4	5.7				% 4.0 (iota)
3=	7.1	14.2	10.4			% 4.2 (virgo omicron)
4=	8.2	11.9	6.9	5.4		% 4.2 (virgo nu)
5=	5.7	9.5	11.4	12.2	13.7	% 4.5 (93)

29. Gienah - 2.8 / corvus gamma

1=	7.2					% 2.8 (beta)
2=	3.6	7.0				% 3.1 (delta)
3=	5.3	5.8	7.7			% 3.2 (epsilon)
4=	7.4	6.1	9.7	2.3		% 4.2 (alpha)
5=	4.2	7.3	0.7	8.3	10.2	% 4.4 (eta)

30. Acrux - 1.1 / crux alpha

1=	4.5					% 1.5 (beta)		
2=	6.2	3.5				% 1.6 (gamma)		
3=	6.2	9.6	12.0			% 2.9 (musca alpha)		
4=	4.7	4.6	2.8	10.6		% 3.1 (delta)		
5=	5.6	8.3	11.2	1.5	10.1	% 3.3 (musca beta)		
6=	6.0	9.6	9.3	8.8	6.7	9.0	% 3.3 (centaurus lambda)	
7=	2.9	3.6	3.7	8.9	2.0	8.1	6.3	% 3.6 (epsilon)
8=	9.0	11.9	14.8	3.3	13.7	3.8	12.0	% 3.6 (musca delta)

31. Gacrux - 1.6 / crux gamma

1=	6.2					% 1.1 (alpha)		
2=	3.5	4.5				% 1.5 (beta)		
3=	8.4	14.2	10.8			% 2.4 (centaurus gamma)		
4=	7.4	12.8	10.9	6.5		% 2.9 (centaurus delta)		
5=	2.8	4.7	4.6	2.8	8.2		% 3.1 (delta)	
6=	9.3	6.0	9.6	17.1	13.2	6.6	% 3.3 (centaurus lambda)	
7=	3.7	2.9	3.6	11.9	9.9	2.0	6.3	% 3.6 (epsilon)

32. Alioth - 1.7 / ursa major epsilon

1=	4.5					ODMBAO
2=	9.0	13.2				% 2.4+4.0+4.0 (zeta)
3=	5.5	9.9	4.5			% 2.5 (gamma)
						% 3.4 (delta)

33. Spica - 1.2 / virgo alpha

1=	6.6					% 4.4 (theta)
2=	*10.7	15.3				% 4.3 (chi)
3=	*10.7	7.5	13.1			% 3.4 (zeta)
4=	5.2	5.2	10.1	5.7		%*4.6
5=	7.4	5.5	17.9	13.0	9.4	%*4.8

34. Alkaid - 1.9 / ursa major nu

1= 6.9
2= 6.7 9.3
3= 5.9 12.0 6.4

ODMBAO
% 2.4+4.0+4.0 (zeta)
% 4.1 (bootes theta)
% 4.3 (bootes lambda)

35. Hadar - 0.9 / centaurus beta

1= 4.8
2= 10.0 14.4
3= 7.7 11.4 9.9
4= 6.6 4.2 14.7 14.2
5= 9.9 5.3 19.9 15.2 7.7
6= 10.0 14.6 2.7 7.8 15.5 19.8

% 0.3+1.7 (alpha)
% 1.5 (crux beta)
% 2.6 (epsilon)
% 3.4 (circinus alpha)
% 4.2 (circinus beta)
% 4.3 (crux mu)

36. Menkent - 2.3 / centaurus theta

1= 7.5
2= 8.3 14.2
3= 6.8 7.8 7.7
4= 6.1 8.1 7.2 0.8
5= 9.7 16.1 13.3 15.8 15.1
6= 6.9 11.4 3.5 4.4 4.0 15.9

% 2.7 (eta)
% 2.9 (iota)
% 3.3 (mu)
% 3.5 (nu)
% 3.5 (hydra pi)
% 4.0 (d)
% (+ 9(4.1-4.4))

37. Arcturus - 0.2 / bootes alpha

1= 4.7
2= 8.4 11.7
3= 6.5 2.0 13.0
*= 10.5 13.9 13.9 15.8

ODMBAO
% 2.8 (eta)
% 4.4 (zeta)
% 4.5 (tau)
% 2.7 (epsilon) (this
value, listed last,
can give negative
brightness values.)

38. Rigel Kentaurus - 0.1 / centaurus alpha

1= 4.8
2= 9.4 13.9
3= 8.9 11.8 6.6
4= 4.2 6.6 8.3 5.3
5= 10.0 12.9 12.8 16.6 13.4
6= 8.5 12.2 3.7 3.0 5.9 14.7
7= 5.4 9.9 6.6 9.9 7.7 6.8 7.9

% 0.9 (beta)
% 3.0 (tri.aust. beta)
% 3.1 (tri.aust. gamma)
% 3.4 (circinus alpha)
% 3.5 (lupus zeta)
% 4.1 (tri.aust. epsilon)
% 4.2 (circinus beta)

39. Zubenelgenubi - 2.9 / libra alpha

1= 9.2
2= 9.6 16.1
3= 7.5 14.0 15.5

% 2.7 (beta)
% 3.4 (sigma)
%*4.6 (virgo lambda)

40. Kochab - 2.2 / ursa minor beta

1=	3.2					% 3.1 (gamma)
2=	5.0	6.2				% 4.3 (zeta)
3=	9.7	11.1	5.1			% 4.4 (epsilon)
4=	2.3	5.4	4.9	8.9		% 4.4 (5)

41. Alphecca - 2.3 / corona borealis alpha

1=	7.9					% 3.5 (bootes delta)
2=	2.9	5.1				% 3.7 (beta)
3=	2.0	9.1	4.5			% 3.9 (gamma)
4=	3.4	10.5	5.8	1.8		% 4.0 (delta)
5=	4.7	4.2	2.6	5.6	1.8	% 4.2 (theta)
6=	9.1	17.0	12.0	8.2	7.9 13.7	% 4.3 (serpens chi)
*=	5.7	12.1	7.8	3.9	2.3 8.0 8.1	%*4.5 :1.5 est, unlisted, not vis.

42. Antares - 1.2 / scorpio alpha

1=	9.1					% 2.4 (epsilon)
2=	7.7	16.0				% 2.5 (delta)
3=	8.7	17.7	3.1			% 2.9 (beta)
4=	2.4	6.9	9.9	11.1		% 2.9 (tau)
5=	6.9	13.8	3.5	6.6	8.5	% 3.0 (pi)
6=	2.1	10.9	5.6	6.9	4.2 5.0	% 3.1 (omicron)
7=	7.8	12.5	6.7	9.7	8.7 3.1 6.6	% 4.0 (rho)
						% (+ omega-1 = 4.1)

43. Atria - 1.9 / triangulum australe alpha

1=	7.9					% 3.0 (beta)
2=	8.2	6.6				% 3.1 (gamma)
3=	7.2	12.3	15.1			% 3.6 (pavo eta)
4=	10.0	8.1	13.9	8.8		% 3.7 (ara eta)
5=	9.6	11.9	16.3	4.6	5.7	% 3.8 (ara delta)
6=	9.9	15.7	11.4	15.1	19.9 18.7	% 3.9 (apus gamma)
7=	6.3	2.4	7.8	10.1	6.4 9.6 15.2	% 4.0 (delta)
						% (+ 2(4.1, 4.2)

44. Sabik - 2.6 / ophiuchus eta

1=	9.5					% 2.7 (zeta)
2=	9.8	17.8				% 3.4 (theta)
3=	6.1	15.0	10.2			% 3.6 (serpens xi)
4=	3.8	10.7	12.1	4.6		% 4.4 (serpens nu)
5=	9.0	17.7	1.3	9.0	11.3	% 4.4 (44)
6=	9.3	6.2	14.5	15.5	12.3 14.7	% 4.4 (phi)
7=	7.4	14.4	12.7	2.7	4.8 11.8 16.7	% 4.4 (serpens omicron)
8=	6.1	14.7	3.9	6.7	8.1 3.4 12.5 9.3	% 4.5 (xi)

45. Shaula - 1.7 / scorpio lambda

1=	5.8								% 2.0 (theta)
2=	9.9	11.7							% 2.0 (sagitar. epsilon)
3=	9.0	12.5	18.3						% 2.4 (epsilon)
4=	2.6	4.1	8.9	11.3					% 2.5 (chi)
5=	0.7	5.9	10.3	8.5	2.8				% 2.8 (upsilon)
6=	8.0	9.7	17.8	3.8	9.5	7.5			% 3.1-3.4 + 3.6 (mu-1+2)
7=	4.0	3.5	8.5	11.4	1.4	10.5	4.3		% 3.1 (iota-1)
8=	9.0	13.1	5.6	15.4	9.4	9.4	16.1	10.1	% 3.1 (sagitar. gamma)
									% (+ 3.2, 3.3, 3.4, ...)

46. Rasalhague - 2.1 / ophiuchus alpha

									ODMBAO
1=	8.6								% 2.9 (beta)
2=	5.2	12.2							% 3.1-3.9 (herc. alpha)
3=	8.0	7.7	13.2						% 3.7 (72)
4=	9.7	12.5	6.7	16.5					% 4.1-5.0 (chi)
5=	9.0	4.2	11.2	11.2	8.9	10.1			% 4.4 (sigma)
*=	10.2	13.5	6.7	17.2	1.1	10.1			% 4.3 (iota)

47. Eltanin - 2.4 / draco gamma

1=	4.0								% 3.0 (beta)
2=	5.5	5.7							% 3.9 (xi)
3=	6.3	6.4	11.2						%*3.9 (hercules iota)

48. Kaus Australis - 2.0 / sagittarius epsilon

1=	9.9								% 1.7 (scorpius lambda)
2=	9.2	2.5							% 2.5 (scorpius chi)
3=	8.9	19.2	18.0						% 2.7 (zeta)
4=	4.6	11.9	11.9	8.5					% 2.8 (delta)
5=	8.9	16.2	16.4	8.4	4.5				% 2.9 (lambda)
6=	5.4	9.5	10.0	11.3	3.1	7.0			% 3.1 (gamma)
7=	9.0	3.8	1.5	17.6	12.1	16.6	10.5		% 3.1 (scorpius iota)
8=	2.7	8.1	6.8	11.2	6.9	11.4	6.6	6.5	% 3.2 (eta)
									% (+2(3.3))

49. Vega - 0.1 / lyra alpha

									ODMBAO
1=	7.7								% 3.3 (gamma)
2=	6.2	2.1							% 3.4-4.3 (beta)
3=	7.8	13.3	11.4						% 4.0 (hercules theta)
4=	6.4	11.6	10.7	12.9					% 4.0-4.5 (R)
5=	4.3	8.7	6.8	4.8	10.3				% 4.3 (chi)
6=	2.1	5.8	4.6	6.4	6.7	5.1			% 4.3 (zeta)
7=	7.1	7.1	7.6	14.9	6.1	10.9	6.0		% 4.5 (eta)
8=	7.9	6.7	7.2	15.5	7.1	11.1	6.1	1.1	% 4.5 (theda)

50. Nunki - 2.1 / sagittarius sigma

1=	3.9								% 2.7 (zeta)
2=	8.1	8.4							% 2.8 (delta)
3=	5.9	8.4	4.7						% 2.9 (lambda)
4=	6.2	8.8	13.5	10.1					% 3.0 (pi)
5=	2.3	4.7	5.9	4.1	8.0				% 3.3 (phi)
6=	2.8	2.4	9.7	8.6	6.6	4.6			% 3.4 (tau)
7=	5.1	8.7	11.7	7.6	2.8	6.5	6.8		% 3.6 (xi-2)
8=	5.0	7.9	12.3	8.6	1.5	6.7	5.9	1.8	% 3.9 (omicron)

51. Altair - 0.9 / aquila alpha

1=	2.1								% 2.8 (gamma)
2=	8.6	9.1							% 3.4 (delta)
3=	7.9	9.7	7.0						% 3.7-4.4 (eta)
4=	9.7	8.0	16.3	17.5					% 3.8 (sagitta delta)
5=	2.8	4.9	8.0	5.3	12.2				% 3.9 (beta)
6=	9.6	7.7	15.3	17.3	2.0	12.2			% 4.4 (sagitta alpha)
7=	8.9	7.0	14.9	16.7	2.0	11.8	0.7		% 4.4 (sagitta beta)

52. Peacock - 2.1 / peacock alpha

1=	9.6								% 3.2 (indus alpha)
2=	9.8	18.6							% 3.6 (beta)
3=	9.7	19.0	3.9						% 3.6 (delta)
4=	4.6	11.7	7.8	9.3					% 3.7 (indus beta)

53. Deneb - 1.3 / cygnus alpha

									ODMBAO
1=	6.1								% 2.3 (gamma)
2=	9.9	8.2							% 3.0 (delta)
3=	9.6	10.4	17.9						% 3.8 (tau)
4=	4.5	8.8	14.0	6.1					% 3.9 (xi)
5=	5.0	6.6	13.6	4.6	3.2				% 4.0 (nu)
6=	4.9	7.3	5.8	14.4	9.3	9.8			% 4.0 (omicron-1)
7=	4.9	6.7	5.1	14.1	9.3	9.6	1.1		% 4.1 (omicron-2)
8=	9.0	13.9	18.9	8.5	5.5	8.0	13.3	13.8	% 4.2 (rho)
9=	8.9	10.7	17.8	1.6	5.0	4.0	13.7	13.7	% 4.3 (sigma)

54. Enif - 2.5 / pegasus epsilon

1=	6.8								% 3.7 (theta)
2=	8.1	12.4							% 4.1 (equuleus alpha)
3=	7.5	12.4	13.7						% 4.5 (9)

55. Al Na'ir - 2.2 / grus alpha

1=	5.7					% 2.2 (beta)
2=	9.7	12.5				% 3.2 (gamma)
3=	7.6	4.6	16.3			% 3.7 (epsilon)
4=	5.0	4.0	8.6	8.3		% 4.0 (delta-1)
5=	9.9	6.5	18.7	2.2	10.5	% 4.2 (zeta)
6=	4.9	3.7	8.9	7.9	0.3 10.1	% 4.3 (delta-2)

56. Formalhaut - 1.3 / piscus austrinus alpha

1=	8.7					% 3.8 (aquarius 88)
2=	4.5	8.5				% 4.2 (epsilon)
3=	2.9	11.7	6.3			% 4.3 (delta)
4=	6.1	13.5	5.7	5.8		% 4.4 (beta)
5=	3.4	12.2	6.3	0.9	5.1	% 4.5 (gamma)
6=	5.1	11.4	9.7	4.7	9.6 5.1	% 4.5 (sculptor gamma)

57. Markab - 2.6 / pegasus alpha

1=	6.9					% 3.6 (zeta)
2=	10.0	13.9				% 3.7 (mu)
3=	9.5	12.9	1.4			% 4.1 (lambda)
4=	5.3	1.9	12.5	11.4		% 4.3 (xi)

APPENDIX C:
POSITION DETERMINATION in DEEP SPACE

This appendix is a brief sketch of a method, and a device to implement the method, of establishing a relative position in space beyond the range of easily identifiable familiar bodies or practical ground support. As stated in the body of this thesis, mankind's instruments are now in this realm of space, where there is nothing to use for positioning except the stars. And while the thrust of this project was to derive a method to autonomously identify the stars for navigational use, this ability does no good in deep space if there is no way to use these now identified bodies. Since a deep-space positioning method was not found in the literature, an idea for one is offered here. I assume the existence of some type of three-dimensional star chart to use as a reference. The coordinate system used can be arbitrary.

What is needed is some structure, based on the stars, that can be used as a reference. Positioning on earth is based on lines-of-position (LOPs) formed between the celestial bodies and the center of the earth. The deep space analogy uses the LOPs that still exist when the earth is removed: between the stars themselves. The basic observation is then that we can establish our location as being on the line between two known bodies whenever they are separated by exactly 180 degrees. The number of these lines

is quite large: if there are N known (or visible) stars available as a reference, then there are N factorial inter-connecting lines between them. Despite the immensity of this number, however, observing two stars to be EXACTLY 180 degrees apart will obviously be a rare occurrence. What is needed, then, is a mechanism that can measure the "nearness" of the spacecraft to one of these stellar LOPs, and a method to use the resulting measurements.

A simple mechanism can be visualized as a long "pipe" or "tube". In the center are two photodetectors, one facing out each open end of the pipe. The ends have a variable aperture that can be made quite small: down to the wavelength of light if possible. The measurement process begins by rotating the pipe with the apertures expanded (open) in three dimensions until a star is detected in each end. Then a two step sequence of

1. reduce the apertures until both stars can no longer be detected, then
2. move the pipe (slightly) in an effort to reacquire the stars

is repeated until the second step is no longer successful. The degree to which the apertures can be reduced and still detect both stars is the measure of the "nearness" to the stellar baseline.

The specific quantity for "nearness" can be seen easily if we first assume the spacecraft is orthogonal to the

center of some baseline, i.e. we are equidistant from both of the stars we are measuring. (For simplification, both photodetectors are considered as "points": they only detect light that is perfectly centered in the tube). Then figure C1 shows that there are two pairs of similar triangles, so that the ratio of the pipe's effective radius "r" (after aperture contraction) to the distance from the stellar baseline ("D") is equal to the ratio of one-half the length of the pipe to one-half the distance between the two stars (total baseline distance). Or,

$$D = \frac{\text{baseline distance} \times \text{aperture radius}}{\text{pipe length}} \quad [C1]$$

when the spacecraft is orthogonal to the baseline center.

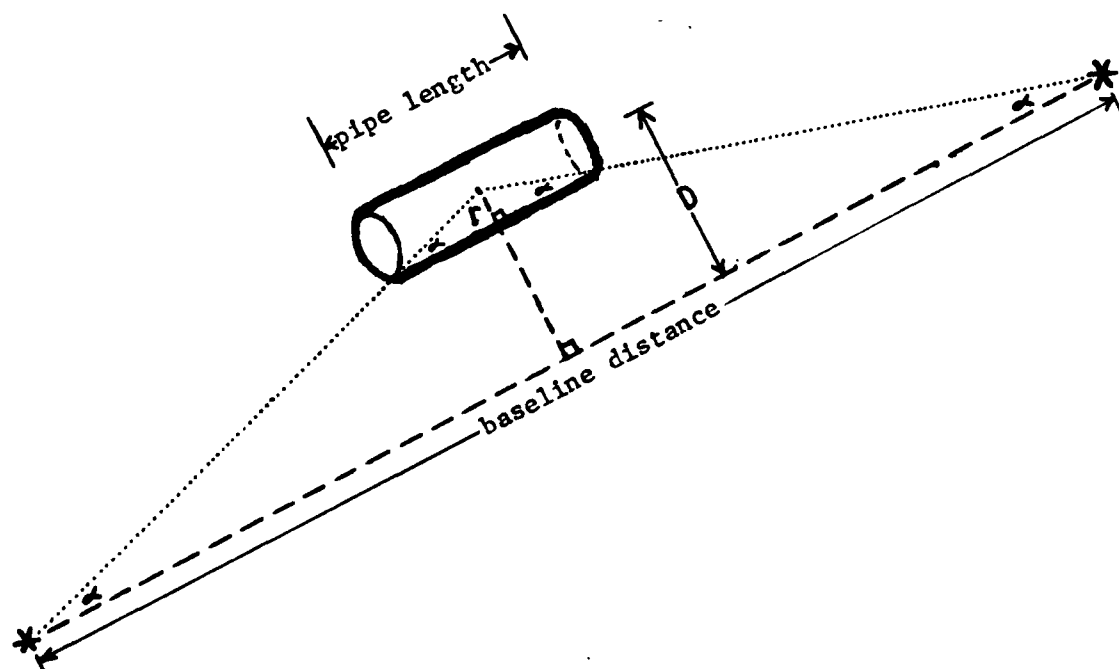


Figure C1: Similar Triangles when Equidistant from Baseline Stars

When the spacecraft is not orthogonal to the center of the baseline, the triangle will no longer be parallel, and a similar degree of aperture closure will indicate a smaller D value (nearer to the baseline) by a factor of the cosine of the distance from the equidistant point ("d" in figure C2). This d value is also equal to the one-half the baseline distance minus the distance to the near star, and the general equation for X, the distance from the stellar baseline from all points, is

$$X = D \times \cos \left\{ \frac{d \times \pi}{\text{baseline distance}} \right\} \quad [C2]$$

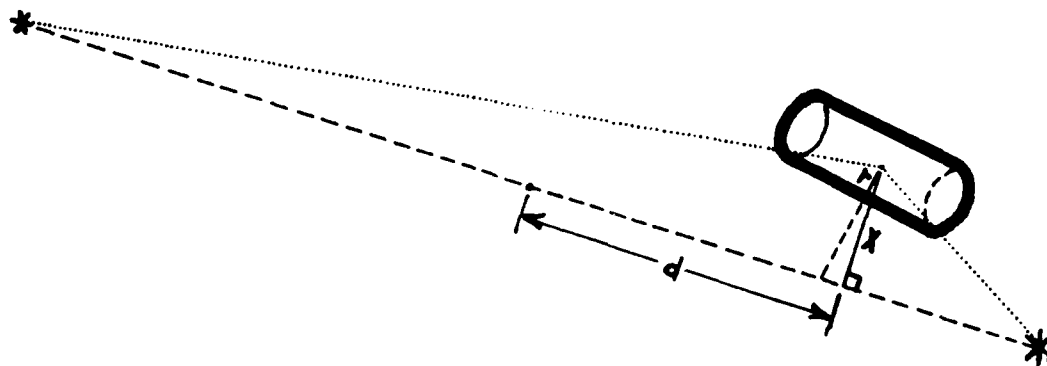


Figure C2: Non-similar Triangles Away from Baseline Center.

Given, then, a measurement of the effective aperture radius "r", the spacecraft's position is somewhere on the surface of a three dimensional sine wave that is constructed around the stellar baseline and defined by the distance X from this baseline. X is a maximum when the pipe is

equidistant from both stars, and approaches zero as the pipe approaches one star or the other. (The spacecraft could be inside the 3-D sine wave, instead of only on its surface, if the apertures reach maximum contraction and both stars are still detectable in the pipe).

Establishing the location of the spacecraft as on the surface of an extremely long sine wave is not, by itself, of much use. But, like on the surface of the earth, the process can be repeated with other star measurements (here star pairs nearly 180 degrees apart) to successively refine a position. Additional measurements would yield a surface of position defined by the intersection of two (or more) of the sinusoidal surfaces. While a closed form solution to this resolution process is not offered here, an iterative process similar to that used in Appendix D would work if the deterministic (non-iterative) equations become unwieldy.

Position Sensitivity

An example gives an indication of the resultant absolute and relative positional accuracy, using a relatively good "pipe", under the fairly poor (for accuracy considerations) conditions of a spacecraft near the midpoint of a fairly long baseline.

Given: two identifiable stars detected in opposite ends of a pipe with an effective total length of 15.43 meters (mirrors should make this very conservative figure) with an aperture that can be reduced to a 2 micron diameter;

the star pair is separated by 500 parsecs, and the previous measurement (which was of a pair roughly orthogonal to this pair) limited the position to near the midpoint of this baseline, or about 250 parsecs to each star.

Then since "d" in equation C2 is approximately zero, the distance from the baseline (X) reduces to D in equation C1. Converting all values to meters, this becomes

$$X = D = \frac{1.543 \times 10^{19} \times 10^{-6}}{15.43} = 10^{12} \text{ meters.}$$

This absolute level of accuracy is of no use for the majority of human endeavors: it is roughly 6 astronomical units, or somewhat greater than the mean radius of Jupiter's orbit around the sun. But outside the solar system boundaries this order of accuracy should be useful, and in relative terms the accuracy is on the order of 10^{-8} of the baseline used, that is, a relative accuracy of ten one-millionths of one percent.

APPENDIX D

This appendix presents an algorithm to iteratively compute an observer's earth-surface coordinates using only the elevation measurements of known celestial objects. The program initially prompts the user for the number of observations to be used to determine the position, however this program has only been tested using three observations per run. A number fewer than three suffers from possible positional ambiguity since each observation provides a locus of position in the form of a circle centered at the sub-point of the body, and two circles will not normally intersect at a single point. More than three observations increase the chances of the system becoming overdetermined, and a single inaccurate observation would prevent convergence to within the requested accuracy. (Exactly three observations may also fail to converge if one or more are in error, since the three loci will not then intersect at a single point).

The algorithm initially assumes a position equal to the arithmetic mean of the sub-points of the bodies: the first latitude is assumed to be at the average of the declinations, and the first longitude at the average sidereal hour angle (SHA). Then the program repeatedly follows the following steps until the change in the latitude and longitude values in successive iterations is less than a prescribed accuracy ("epsilon"):

1. Compute the height and azimuth of each body from the current assumed position.
2. Determine the amount of error between this computed height and the actual height observed for each body. Retain the maximum of these errors.
3. Determine a new assumed position that zeros-out the maximum error, by computing a dead-reckoning position based on a displacement from the last assumed position equal to the maximum error amount and along the line of the computed azimuth.

Since the initial assumed position is at an average latitude and longitude, choosing bodies that surround an observer will normally result in a faster convergence than if the observed bodies all are in a similar direction or region of the sky. However the algorithm still converged in this instance for all of the cases tested.

Using only a single observational error to compute the next assumed position is used because it was found to give a faster convergence than any scheme that used some weighted average of all of the observational errors. The program will output the number of iterations needed to converge to the desired precision, which is currently set to 1.0 nautical mile.

```

program resolvefixwithoutazimuth(input,output);
var
    sha,
    dec,
    ho,
    hc,
    zn,
    drdistance                      : array [1..5] of real;

    corr, c45,
    pi, epsilon,
    distance,
    newlat, newlong,
    assumedlat, assumedlong        : real;

    finished                        : boolean;

    numberofobs,
    index,
    iterationcount,
    i                               : integer;

(* ===== trig functions not on the Vax ===== *)
function tan(x : real) : real;
begin
    tan := ( sin(x) / cos(x) );
end;

(* ----- *)
function arccos(x : real) : real;
var y,z      : real;
begin
    z := abs(x);
    if x > 0 then
        y := arctan ( (sqrt(1 - (x*x) ) ) / x)
    else
        y := pi - arctan ( (sqrt(1 - (z*z) ) ) / z);
    if y > (2*pi) then y := y - (2*pi);
    arccos := y;
end;

(* ===== *)

```

begin

```
pi := 3.14159265359;
corr := pi / 180; (* degrees to radians *)
epsilon := 1.0; (* desired accuracy of procedure *)
write ('Enter the number of observations to be used ');
writeln('to determine the position. ');
readln(numberofobs);
writeln('Enter (sha, dec, ho ) * ', numberOfobs);
for i := 1 to numberOfobs do begin
    readln(sha[i], dec[i], ho[i]);
    sha[i] := sha[i] * corr;
    dec[i] := dec[i] * corr;
    ho[i] := ho[i] * 60;
end;

for i := 1 to numberOfobs do begin
    assumedlat := assumedlat + dec[i];
    assumedlong := assumedlong + sha[i];
end;

(* compute initial wag position: *)
assumedlat := assumedlat / numberOfobs;
assumedlong := assumedlong / numberOfobs;
writeln('asslat1: ', assumedlat, ', asslong1: ',
        assumedlong);
```

```
while not finished do begin
    for i := 1 to numberOfobs do begin
        (* ----- compute Hc's and Zn's ----- *)
        distance := arccos( (sin(assumedlat) * sin(dec[i]))
            + ( cos(assumedlat) * cos(dec[i]) *
                cos (sha[i] - assumedlong) ) ) );
        zn[i] := arccos( (sin(dec[i]) -
            (sin(assumedlat) * cos(distance) ) )
            / (sin(distance) * cos(assumedlat) ) );
        distance := 60 * (distance/corr) ; (* back to nm *)
        hc[i] := 5400 - distance; (* in base 10 *)
        zn[i] := zn[i] / corr;
        if sha[i] > assumedlong then
            if ( (zn[i] > 0) and (zn[i] < 180) ) then
                (* resolve quadrant *)
                zn[i] := 360 - zn[i];
        if sha[i] < assumedlong then
            if ( (zn[i] < 0) or (zn[i] > 180) ) then
                zn[i] := 360 - zn[i];
```

```

(* ----- compute DR distance and direction ----- *)
if ho[i] < hc[i] then
  zn[i] := zn[i] + 180;      (* Vax uses degrees *)
if zn[i] < 0 then
  zn[i] := zn[i] + 360;
if zn[i] >= 360 then
  zn[i] := zn[i] - 360;
zn[i] := zn[i] * corr; (* back to radians for ln *)
drdistance[i] := abs(ho[i] - hc[i]);
end (* of for-loop *) ;

index := 1;
for i:= 2 to numberofobs do
  if drdistance[i] > drdistance[index] then
    index := i;
    (* index now refers to the maximum error body *)

(* ----- compute next assumed position ----- *)
writeln;
writeln('index: ', index, ' ', drdistance[index],
        ' / ', zn[index]);
if drdistance[index] < epsilon then
  finished := true;
newlat := (assumedlat/corr) +
           ( (drdistance[index]/60) * cos(zn[index]) );
if ( (abs(zn[index]) = 90) or (abs(zn[index]) = 270) )
  then writeln('Zn exactly 090 or 270 degrees. ');
c45 := 45 * corr;
assumedlong := assumedlong/corr +
               ( (tan(zn[index]) / corr) *
                 ln( (tan( (assumedlat/2) + c45) )
                   / (tan( (newlat*corr/2) + c45) ) ) );
(* alternate termination test:
if ( (abs(newlong - assumedlong) < epsilon) and
    (abs(newlat - assumedlat) < epsilon) ) then
  finished := true;
assumedlong := newlong; *)

assumedlat := newlat;
writeln('new asslat: ', assumedlat,
        ' ', new asslong: ', assumedlong);
writeln;
assumedlat := assumedlat * corr;
assumedlong := assumedlong * corr;
iterationcount := iterationcount +1;
end (* of the while-loop *) ;

writeln;
writeln('Program used', iterationcount,
        ' iterations for an accuracy of ', epsilon);

```

end.

APPENDIX E: GLOSSARY

A Brief Glossary of Terms Specific to this Project.

candidate: 1. in general, a star (usually the brightest body in a sector or in the entire field-of-view) that the pattern recognition program will try to identify, by determining which template in the database most closely resembles the sensor image that includes this star. 2. in the main computer program, this is the name of the procedure that is called to perform quadrilateral-matching and brightness-correlation tests after the first hurdle (a successful triangle match) has been achieved with one of the templates.

catalog-star: one of the 57 stars that are catalogued in the database as the pattern-recognition templates.

confidence ratio: for any sample, the ratio of the highest point total accrued by any of the catalog-stars to the second highest point total; the point total for the "selected" template divided by the highest point total of all other (non-selected) templates. It is therefore a measure of how much the selected template "stands out" from the rest.

higher-order polygon: a polygon with a greater number of vertices. (It would therefore also have an increased number of sides, but this project defines the geometric structures by their number of vertices).

nested recomposition: the recursive method used in this thesis to perform the pattern-recognition of variable length sample vectors with variable length template vectors. Single sides, then triangles, then higher-order polygons are successively computed, then compared to the template set for a match, then linked with other structures of the same type to build the next larger structure, i.e. single side lengths from a sample are compared to template values in an attempt to build matched triangles, then the triangles are compared in an attempt to form quadrilaterals, and the process can be repeated as long as is necessary.

quadrilateral: any polygon with sides formed by lines adjoining exactly four vertices. (This differs slightly from the standard geometric definition because a quadrilateral, as defined here, will have a total of six sides counting the internal "sides").

rich-field star: a catalog-star (or candidate-star) with a relatively larger number of support-stars of sufficient brightness within a usable distance. For the normal parameters used in the pattern recognition program, a catalog (candidate) star with six or more other stars of the fourth magnitude (or brighter) and within a ten degree radius.

sparse-field star: a catalog-star (or candidate-star) with a lesser number of usable support-stars. Generally four or fewer stars of at least fourth magnitude brightness with ten degrees of the catalog (candidate) star in this project.

subpoint (of a celestial body): standard navigation terminology, used to denote the unique point on a sphere (commonly the surface of the earth) where, at a given instant of time, the vector to the celestial body is orthogonal to the plane of the horizon; the point where the body is at the zenith.

support-radius (also called "inclusion radius") : a distance (less than one half of the sensor's effective field-of-view) used to define the support-stars for a catalog-star or a potential candidate-star. Only the support-stars within the support-radius of the catalog-star are stored in the data base, and only the support-stars within the support-radius of a candidate-star are processed by the recognition algorithms.

support-star: other stars, generally less bright than a catalog-star (or candidate-star), used to define the two-dimensional polygons chosen as the primary feature set used for star identification.

template: one of the 57 catalog-stars with its feature vector composed of the distances between it and its support-stars and between the support-stars themselves.

VITA

Eugene P. Schempp was born on 6 December 1952 in Englewood, New Jersey. He graduated from Parsippany Hills High School, Parsippany, New Jersey, in 1971. In June 1975 he received the degree of Bachelor of Science, majoring in Computer Science, and a commission as a Second Lieutenant from the USAF Academy. He became a rated navigator upon completing Undergraduate Navigator Training in May 1976. He was then assigned to the 41st Air Refueling Squadron, Griffiss AFB, New York, where he served as a KC-135 navigator, instructor navigator, and the Training Flight/Upgrade instructor navigator. He received the degree of Master of Science in Management Science, majoring in Systems Management, from the State University of New York at Binghamton in May 1981. He entered the AFIT School of Engineering in June 1982.

Permanent address: 75 Alexander Avenue
Parsippany, New Jersey 07054

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

4188115

1. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/EE/83D-19			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION School of Engineering		6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433			7b. ADDRESS (City, State and ZIP Code)	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AF Space Technology Center		8b. OFFICE SYMBOL (If applicable) Det 1, YLXS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State and ZIP Code) PO Box 92960, Worldway Postal Center Los Angeles, Ca. 90009			10. SOURCE OF FUNDING NOS.	
11. TITLE (Include Security Classification) See block 19.			PROGRAM ELEMENT NO.	PROJECT NO.
			TASK NO.	WORK UNIT NO.
12. PERSONAL AUTHOR(S) Eugene P. Schempp, Capt, USAF				
13a. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1983 December	15. PAGE COUNT 150
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB. GR.	Celestial Pattern Recognition, Star Identification, Autonomous Navigation	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)				
Title. CELESTIAL PATTERN RECOGNITION ALLOWING AUTONOMOUS EARTH-SURFACE OR DEEP-SPACE POSITIONING. Abstract: This Project explores the feasibility of using a small general digital processor, with an optical sensor and a clock, as a means to autonomously determine location. A stellar pattern-recognition algorithm is presented that determines the identity of any of the navigation stars catalogued in the U.S. Naval Observatory <u>Air Almanac</u> which may be in the sensor's field of view. Appendices describe an iterative method for determining position on the surface of the earth without using azimuth information, and a mechanism for determining relative position in deep space. Thesis Chairman: Matthew J. Kabrisky, Ph.D.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Matthew Kabrisky			22b. TELEPHONE NUMBER (Include Area Code) 513-255-5276	22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

END

FILMED

300

DTIC