

LEVEL

1078809

(12) 52

DDIC
ELECTE
DEC 10 1980

AD A092875

A Final Report

Contract No. N0014-78-C-0695

DISCRETE ANALOG PROCESSING FOR
TRACKING AND GUIDANCE CONTROL

Submitted by:

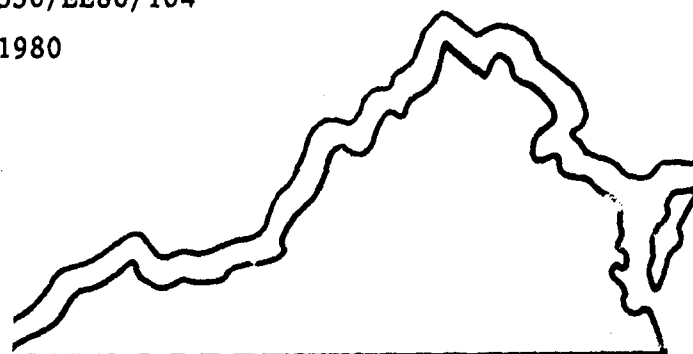
E. S. McVey
Professor

E. A. Parrish
Professor

R. M. Inigo
Associate Professor

Report No. UVA/525350/EE80/104

November 1980



DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DDIC FILE COPY

DDIC FILE COPY



COMPUTER AND CONTROL LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCES
UNIVERSITY OF VIRGINIA



**Best
Available
Copy**

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) ONR-CR-17-092-2F	2. GOVT ACCESSION NO. AD-A092 875	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (16) DISCRETE ANALOG PROCESSING FOR TRACKING AND GUIDANCE CONTROL		5. TYPE OF REPORT & PERIOD COVERED Final Report: 8/1/78-1/15/81
7. AUTHOR(s) (10) E. S./McVey R. M./Inigo E. A./Parrish		6. PERFORMING ORG. REPORT NUMBER (14) UVA/525350/EE80/104
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering New School of Engineering and Applied Science University of Virginia Charlottesville, VA 22901		8. CONTRACT OR GRANT NUMBER(s) (13) N0014-78-C-0695
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 N. Quincy Street Arlington, VA 22217		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (12) 287 C
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (9) Final Report 2 Aug 78-15 Jan 81 N/A		12. REPORT DATE November 1980
		13. NUMBER OF PAGES 174
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) N/A		
18. SUPPLEMENTARY NOTES N/A		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) real-time Model tracking Taylor series algorithm texture affine-transformation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Implementation of the Taylor Series Video Image Processing (TSVIP) algorithm for tracking and guidance is presented using an HP 2100 digital computer to solve the equation. Closed loop system operation was achieved using a Reticon 100 x 100 matrix photodiode camera, a microprocessor controlled A/D converter as an interference to supply signals to the computer and pan tilt servos operating from computer generated signals to control camera position. Tentative studies of CCD implementation including experimental data are presented for eventual elimination of the digital computer. — Sweet Ray		

20. (continued)

Alternate algorithm studies compare the capabilities of complementing and supplementing methods to the TSVIP. A hierarchy of algorithms is anticipated for eventual practical implementation.

A Final Report
Contract No. N0014-78-C-0695
DISCRETE ANALOG PROCESSING FOR
TRACKING AND GUIDANCE CONTROL

Submitted to:
Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217

Submitted by:
E. S. McVey
Professor
E. A. Parrish
Professor
R. M. Inigo
Associate Professor

Department of Electrical Engineering
RESEARCH LABORATORIES FOR THE ENGINEERING SCIENCES
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA

Report No. UVA/525350/EE80/104
November 1980

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

Copy No. 21

TABLE OF CONTENTS

	<u>Page</u>
LIST OF TABLES	vi
LIST OF ILLUSTRATIONS	vii
ABSTRACT	ix
<u>Chapter</u>	
I. INTRODUCTION	1
1.1 Background	1
II. EXPERIMENTAL SYSTEM	4
2.1 System Description	4
2.2 Hardware Drivers	4
2.2.1 Pan Tilt Driver	10
2.2.2 Camera Driver	12
2.3 Video Interface	17
2.4 Tracking	26
2.5 Open Loop Tracking	27
2.6 Closed Loop Tracking	27
III. CCD IMPLEMENTATION	33
3.1 Introduction	33
3.2 Sum of Products (SOP) Matrix Manipulator	34
3.3 Digital/Analog Matrix Manipulator	37
3.3.1 A Programmable Digital/Analog Correlator	37
3.3.2 Use of the Programmable Digital/Analog Correlator for Matrix Operations	39
3.3.3 Use of the Digital/Analog Correlator for the Implementation of the TSVIP	41

TABLE OF CONTENTS

(Continued)

	<u>Page</u>
3.4 MOS Analog Multiplier Compatible with CCD Structures	47
3.5 Experimental Results for the Multichip Implementation of $[P_c]^+$	50
3.5.1 Introduction	50
3.5.2 Experimental Circuit	51
3.5.3 Conclusions and Recommendations	61
IV. THE USE OF SEQUENTIAL ESTIMATION TECHNIQUES IN IMPLEMENTING THE TSVIP ALGORITHM	63
4.1 Introduction	63
4.1.1 Background	63
4.1.2 Closed Form Techniques	63
4.1.3 Sequential Procedures	64
4.2 Minimum Squared-Error Descent Procedures	65
4.2.1 Introduction	65
4.2.2 Multi-Sample Algorithm	66
4.2.3 The Widrow Hoff Rule	67
4.2.4 Convergence Acceleration	68
4.2.5 Startup	70
4.3 Simulation	70
4.3.1 Introduction	70
4.3.2 Convergence Acceleration	71
4.3.3 Size of Perturbation	71
4.3.4 Startup	79
4.3.5 Noise Susceptibility	81
4.3.6 Summation	81

TABLE OF CONTENTS
(Continued)

	<u>Page</u>
V. OTHER ALGORITHMS	84
5.1 Introduction	84
5.1.1 Purpose	84
5.1.2 Constraints	84
5.1.3 Assumptions	86
5.2 The Algorithms	88
5.2.1 Gradient Algorithms	88
5.2.1.1 Development	88
5.2.1.2 Assumptions	91
5.2.2 Moment Algorithms	92
5.2.2.1 Development	92
5.2.2.2 Assumptions	94
5.2.3 Coordinate Algorithms	94
5.2.3.1 Derivation	94
5.2.3.2 Assumptions	95
5.2.4 A Kalman Based Solution of the Affine Parameters . . .	96
5.2.5 Transformation Algorithms	98
5.2.5.1 Derivation	98
5.2.5.2 Assumptions	101
5.2.6 Segmentation Algorithms	102
5.2.6.1 Intensity Segmentation	102
5.2.6.2 Texture Segmentation	103
5.2.6.3 Gestalt Segmentation	104
5.2.6.4 Principal Axis Parameter Estimation	104
5.2.6.5 Projection Parameter Estimation	105

TABLE OF CONTENTS

(Continued)

	<u>Page</u>
5.2.6.6 Assumptions	105
5.3 Implementation	106
5.3.1 Imager	108
5.3.2 Preprocessor	108
5.3.3 Moment Estimator	110
5.3.4 Parameter Estimator	113
5.3.5 Imager Controller	117
5.4 Summary	117
VI. RECOMMENDED FUTURE RESEARCH	121
BIBLIOGRAPHY	123
 <u>Appendices</u>	
2.1 Pin Connections for Video Interface Cable	A-1
2.2 Photographs of Tracking System	A-2
2.3 Software for Tracking System	A-4
2.4 Sequence of Segmented Tracking Scenes	A-48
2.5 Parts Layout for Video Interface	A-50

LIST OF TABLES

<u>Table</u>		<u>Page</u>
5.1	Algorithm Comparison Chart	120

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
2.1	Video Tracking System Example	5
2.2	I/O Interface to HP 2100	8
2.3	Block Diagram of HP 2100 I/O interface	9
2.4	Pan Tilt Mount Hardware Control Codes	11
2.5	Flowchart of Pan Tilt Driver	13
2.6	Block Diagram of HP 2100 Non-interrupt Data Transfers	14
2.7	Block Diagram of HP 2100 Direct Memory Access	15
2.8	Digital Subsection of Video Interface	20
2.9	Analog Subsection of Video Interface	21
2.10	Block Diagram of Digital Subsection of Video Interface . . .	22
2.11	Open Loop Data (each point averaged ten times).	28
2.12	Open Loop Data (no averaging)	29
2.13	Open Loop Data (each point averaged 100 times)	30
3.1	Basic SOP Cell	35
3.2	The Implementation of $[P_c]^+$ Using CCDs and Analog Multiplier	38
3.3	A Programmable Digital/Analog Correlator	40
3.4	System for Computing $M_x = \sum_{i=-m}^m \sum_{j=-n}^n p'(i,g)x_i$	42
3.5	Digital/Analog Implementation of Element c_{11} of matrix $[P_c]^+$	44
3.6	All Parallel Digital/Analog Processor (timing not indicated)	46
3.7	nMOS Multiplier Basic Cell	48
3.8(a)	Output at Tap 1 and Input, SIPO Device	52
3.8(b)	Sinusoids of Fig. 3.8(a) with a Much Shorter Sweep Time . . .	52
3.9	PISO Devices Circuit Diagram	53

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
3.10 Multipliers and Summers Circuit Diagram	55
3.11 SIPO Devices Circuit Diagram	56
3.12(a) ϕ_T and ϕ_2 Waveforms 5V/cm; Sweep at 0.5 us/cm	58
3.12(b) ϕ_T Clock and Output of PISO Device No. 1	58
3.13(a) PISO Devices Output	59
3.13(b) First Two Multipliers Output	59
3.14(a) Output of Summers	60
3.14(b) SIPO Devices Output at Taps No. 9	60
4.1 CCD Implementation of Widrow-Hoff Rule	69
4.2 Flowchart of Implementation of Widrow-Hoff Rule	72-73
4.3 Trans. Vector vs. # of Iterations (no dilation or rotation) .	75
4.4 Trans. Vector vs. # of Iterations (no dilation or rotation) .	76
4.5 Trans. Vector vs. # of Iterations (no dilation or rotation) .	77
4.6 Trans. Vector vs. # of Iterations (no dilation or rotation) .	78
4.7 Angle of Rotation vs. # of Iterations and Dilation vs. # of Iterations	80 82
4.8 Trans. Vector vs. # of Iterations W/Added Noise	
5.1 Example System Configuration	107
5.2 Imager with Data Readout Sequence	109
5.3 Preprocessor with Normalization and Storage	111
5.4 Relationship Between Coordinates and Iteration Index, k, of a 32 by 32 Subimage	112
5.5 Moment Estimator	114
5.6 Digital by Analog Multipliers for Moment Estimator	115
5.7 Moment Estimator Output	116
5.8 Digital Kalman Parameter Estimator	118

ABSTRACT

Implementation of the Taylor Series Video Image Processing (TSVIP) algorithm for tracking and guidance is presented using an HP 2100 digital computer to solve the equation. Closed loop system operation was achieved using a Reticon 100 x 100 matrix photodiode camera, a microprocessor controlled A/D converter as an interface to supply signals to the computer and pan tilt servoes operating from computer generated signals to control camera position. Tentative studies of CCD implementation including experimental data are presented for eventual elimination of the digital computer. Alternate algorithm studies compare the capabilities of complementing and supplementing methods to the TSVIP. A hierarchy of algorithms is anticipated for eventual practical implementation.

Chapter I

INTRODUCTION

1.1 Background

This report describes the second year of research concerning the fundamentals of tracking and guidance using computer vision with an emphasis on Charge Coupled Device (CCD) discrete analog signal processing to achieve real time operation with small rugged and portable hardware. This choice of signal processing technology provides the possibility of having the image sensor and processing circuitry on the same substrate, thereby minimizing the interfacing problems between two major sections of the system. The discrete analog nature of the circuitry further eliminates A/D and D/A hardware. Refer to final report number ONR-CR233-092-1 for a complete description of research results for the first year.

Basically, the idea of the system is to compare successive frames of computer vision data to compute relative motion and displacements in three dimensions. This information is then used to derive guidance and/or pointing signals to keep the desired target in the center of the field of view. A major concern of the research has been the problem of processing rapidly the very large quantities of data required for effective real time computer vision tracking. See Chapter II and last year's report for a complete description of the system and its mathematical representation.

The Taylor Series Video Image Processing (TSVIP) algorithm was developed because an adequate algorithm could not be found in the literature that appeared to be CCD implementable. The basic tracking algorithm should be very fast, but it need not necessarily contain pattern recognition ability, registration ability, edging ability, etc. All these auxiliary operations can be obtained at a relatively slow rate if the tracking algorithm can keep the system functioning while they are given time to function. And, they probably will eventually be required because target discrimination with a very high level of confidence is a major goal.

The TSVIP operates on the gradient of the reflected target light intensity, i.e., the texture. If the texture function at a point on the target is described using a Taylor series and then truncated, a numerical method is available (called the Euler method) for extrapolating texture values. However, if the formula is solved for displacement of the target from the reference point, then texture measurements from successive frames of data yield calculated values of translation. If multiple points are measured and the problem formulated to include the affine transformation, dilation and rotation can be estimated as well as translation in two dimensions. The complete formulation is described in the previous report.

The major emphasis this year has been on implementation to validate system concepts and signal processing technology. Since CCDs are specialized devices, only a limited amount of experimental circuit work has been possible, but this was known beforehand to be the case. However, this work has presented many insights and given credence to the theoretical implementation work. An HP 2100 digital computer with associated components was used for the experimental system and is described in Chapter II. The CCD implementation work is described in Chapter III.

Only one method was considered during the first year for the required pseudoinverse described in Chapter II. Iterative methods are reported in Chapter IV for solving the estimation equations, i.e., obtaining pseudoinverse results. Chapter V contains advanced algorithm work in which the TSVIP and other algorithms are compared. Conclusions are written into each section because of the diverse nature of the individual parts. Chapter VI is concerned with recommended future work.

Three papers have been written on the results to date:

1. "Algorithm Development for Real-Time Automatic Video Tracking Systems " COMPSAC Proceedings (November 1979) (invited paper).

2. "A Model and Tracking Algorithm for a Class of Video Targets," accepted by IEEE Transactions on Pattern Analysis and Machine Intelligence (1980).

3. "CCD Implementation of a Novel Video-Tracking Algorithm," accepted by IEEE Transactions on Pattern Analysis and Machine Intelligence (1980).

It is anticipated that at least three more papers will be published on the recent results of the research.

The second year goal of implementing closed loop equations was achieved.

CHAPTER II

EXPERIMENTAL SYSTEM

2.1 System Description

As already noted an experimental system was constructed to test operation of the TSVIP algorithm and in general lend credence to the theoretical work being done. A block diagram of the system is shown in Fig. 2.1. A Reticon MC/RS 520 camera [1] system acquires data from the target in the form of variation in light intensity or texture and outputs a video signal to an interface. The interface converts this information into a digital signal and latches it into the computer. The computer simulates the CCD portion of the TSVIP algorithm. It outputs signals to control a pan tilt mount in two dimensions although operation in only one dimension was performed. This pan tilt mount responds to the positional control signals to keep the target centered in the camera field of view.

Developing such an experimental system is a major undertaking even though a digital computer is used to simulate part of the hardware. Closed loop operation in two dimensions was achieved. Slow delivery of parts and typical hardware problems such as noise and faulty components were encountered as one normally expects.

2.2 Hardware Drivers

A device driver is the software necessary to interface the computer with the actual hardware. It provides the overall system control and the communication between the hardware and the computer. All the software drivers were written in HP assembler language in

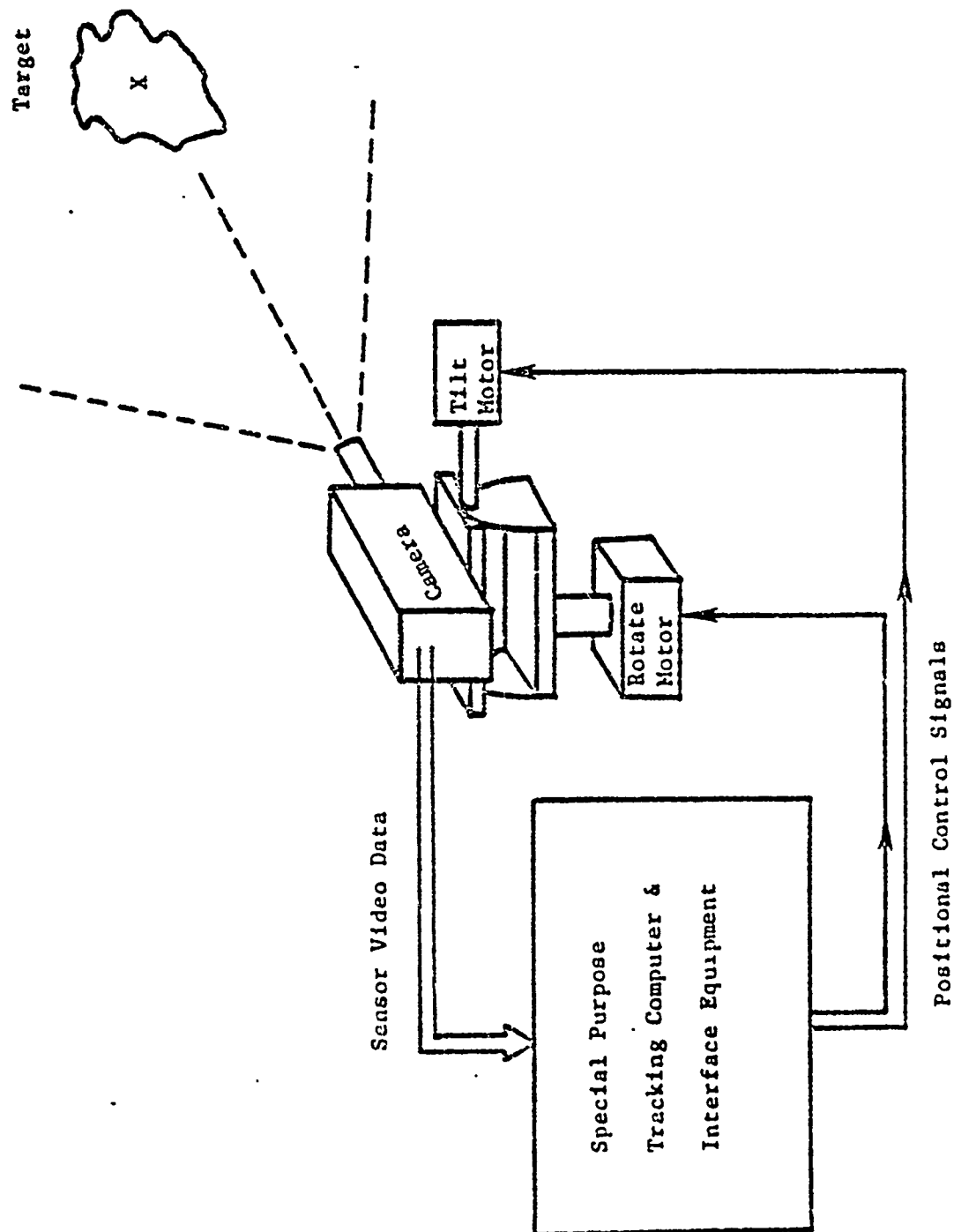


Figure 2.1 Video Tracking System Example

order to be compatible with the existing system.

In order to understand the operation of the device drivers, it is necessary to present a brief description of the facilities. The tracking simulation and emulation has been done in the Control and Computer Systems Laboratory. Major items of equipment used are:

- 1) HP 2100A mini computer
- 2) HP magnetic tape drive
- 3) HP disc drive
- 4) Reticon TV camera
- 5) HP X-Y display
- 6) Tektronix graphics terminal
- 7) GE Terminet
- 8) High speed tape punch and optical tape reader
- 9) Pan tilt mount
- 10) Computer controlled motorized arm

The HP 2100 has a core memory size of 32K and can run under Disc Operating System (DOS) or Binary Control System (BCS), the former residing on the disc and the latter, loaded directly into core memory [2]. Since most developmental programs were written in Fortran (with the exception of the drivers), and because DOS allows relatively easy, fast editing, compilation, loading, and execution of routines in both Fortran and HP Assembler, DOS was used exclusively.

Rectangular pulses are generated and sent to the drivers by the HP 2100A through an I/O channel. The HP 2100A is a mini computer featuring a relatively strong instruction set, plug in interfaces and modular software. Standard features include memory parity generation

and checking, memory and I/O protect for executive systems, extended arithmetic capability, and power fail interrupt with automatic restart. It has a 16 bit word length, 980 nanosecond cycle time, and 80 basic instructions.

Interfacing of peripheral devices is accomplished by plug-in interface cards (see Fig. 2.2). The computer mainframe can accommodate up to 14 interface cards and up to 45 with the I/O extender added. All I/O channels are buffered and bidirectional, and are serviced through a multilevel priority interrupt structure.

The 17 I/O instructions provide the capability to set or clear the I/O flag bits, and to transfer data between an I/O channel and the A or B registers. The general purpose of the I/O system is to transfer data between the computer and external devices. Normally data is transferred through the A or B registers. This type of transfer occurs in three distinct steps:

- 1) between external device and its interface card in the computer
- 2) between the interface card and the A or B registers
- 3) between the A or B registers and memory.

This three step process applies to data following both in and out of the memory. The basic block diagram is shown in Fig. 2.3.

Hardware controllers are connected through buffers by a cable directly to an interface card inside the computer. The interface card in turn plugs into one of the 14 I/O slots. Each slot is assigned a fixed address, called a select code. The computer can then communicate with the device on the basis of its select code (see Fig. 2.1).

For this system, interface cards were installed in the slots

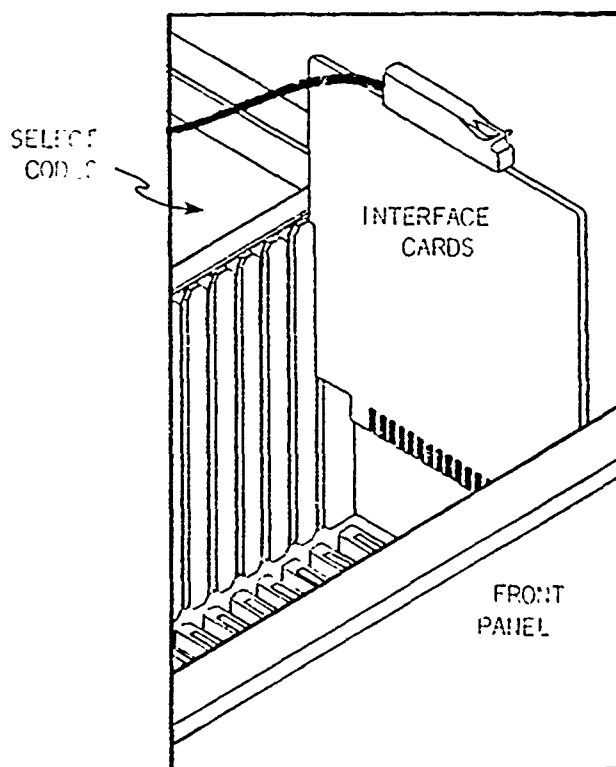
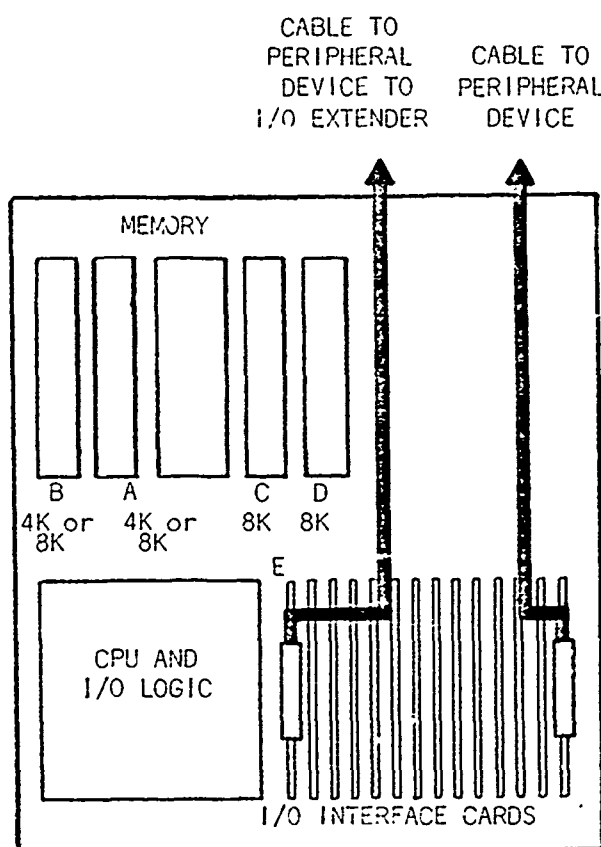


Figure 2.2 I/O Interface to HP 2100

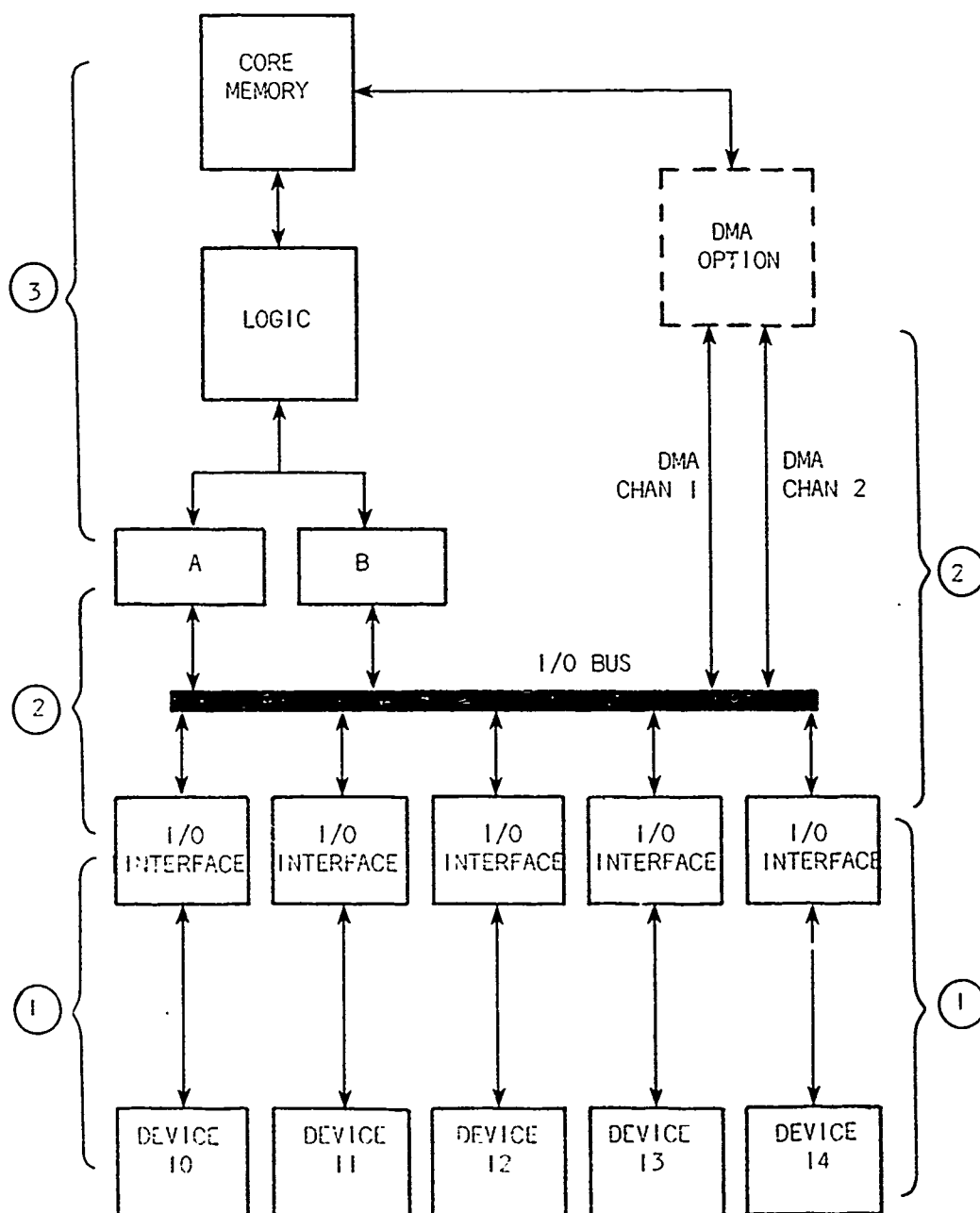


Figure 2.3 Block Diagram of HP 2100 I/O Interface

corresponding to select codes 16 and 36. Data to be transferred are loaded into the register and are a suitable form for output. The operation begins with a HP assembly language program instruction to transfer the data from the A register to the interface buffer. The buffer is a flip-flop register for the intermediate storage of data and its data capacity is 16 bits. Since this is a non-interrupt transfer, and control and flag flip-flops are not used, the sixteen bits of binary data are immediately transferred to the external devices, where a binary zero corresponds to zero volts and a binary one corresponds to five volts. Pin connections for these interface cards are shown in Appendix 2.1.

2.2.1 Pan Tilt Driver

A binary one can be stored in any one of bits twelve to fifteen (depending on the desired movement of the pan and tilt motors), causing that bit to be high and output five volts to the corresponding input terminal of the driver hardware [3,4]. Five different control words are output periodically, one at a time, depending on which motor is to be operated and in which direction it is to be moved. Figure 2.4 lists the code used for the control words. Rectangular pulses are sent to the desired input of the driver by outputting one of these four control words or 0 periodically in order that the desired motor move the required number of steps.

The assembler subroutine is called by this FORTRAN subroutine call:

```
CALL MOTOR(ISTEP1, ISTEP2, IDIR, IPER)
```

where

Code Used for the Data

Data (Octal)	Bit #
10000	Bit 12 is high
20000	Bit 13 is high
40000	Bit 14 is high
100000	Bit 15 is high
0	All bits are low

Bits Used to Output the Data

Bit #	Motor Driver 1	Motor Driver 2
12	Forward	—
13	Reverse	—
14	—	Forward
15	—	Reverse

Figure 2.4 Pan Tilt Mount Hardware Control Codes

ISTEP1 is the number of steps to be taken by the tilt motor

ISTEP2 is the number of steps to be taken by the pan motor

IDIR is the direction of rotation

0 for CCW tilt and CCW pan rotation

1 for CW tilt and CCW pan rotation

2 for CCW tilt and CW pan rotation

3 for CW tilt and CW pan rotation

IPER is the speed of the motors

1 for 20 steps/second

2 for 100 steps/second

3 for 200 steps/second

A simple flow chart of the program is illustrated in Fig. 2.5. The actual assembler program is listed in Appendix 2.3.

2.2.2 Camera Driver

The driver for the camera controller utilizes non-interrupt data transfers and direct memory access, see Figs. 2.6 and 2.7 (several of the figures are similar to information from [2,3]). In a manner similar to the pan tilt mount, the computer outputs the control words to the camera controller by a non-interrupt data transfer; but, because the data rate of the camera output is faster than that of a non-interrupt transfer it was necessary to directly transfer the data into memory.

As already mentioned and as shown in Fig. 2.7, the purpose of the direct memory access (DMA) is to provide a direct data path which is software assignable between memory and the high speed peripheral

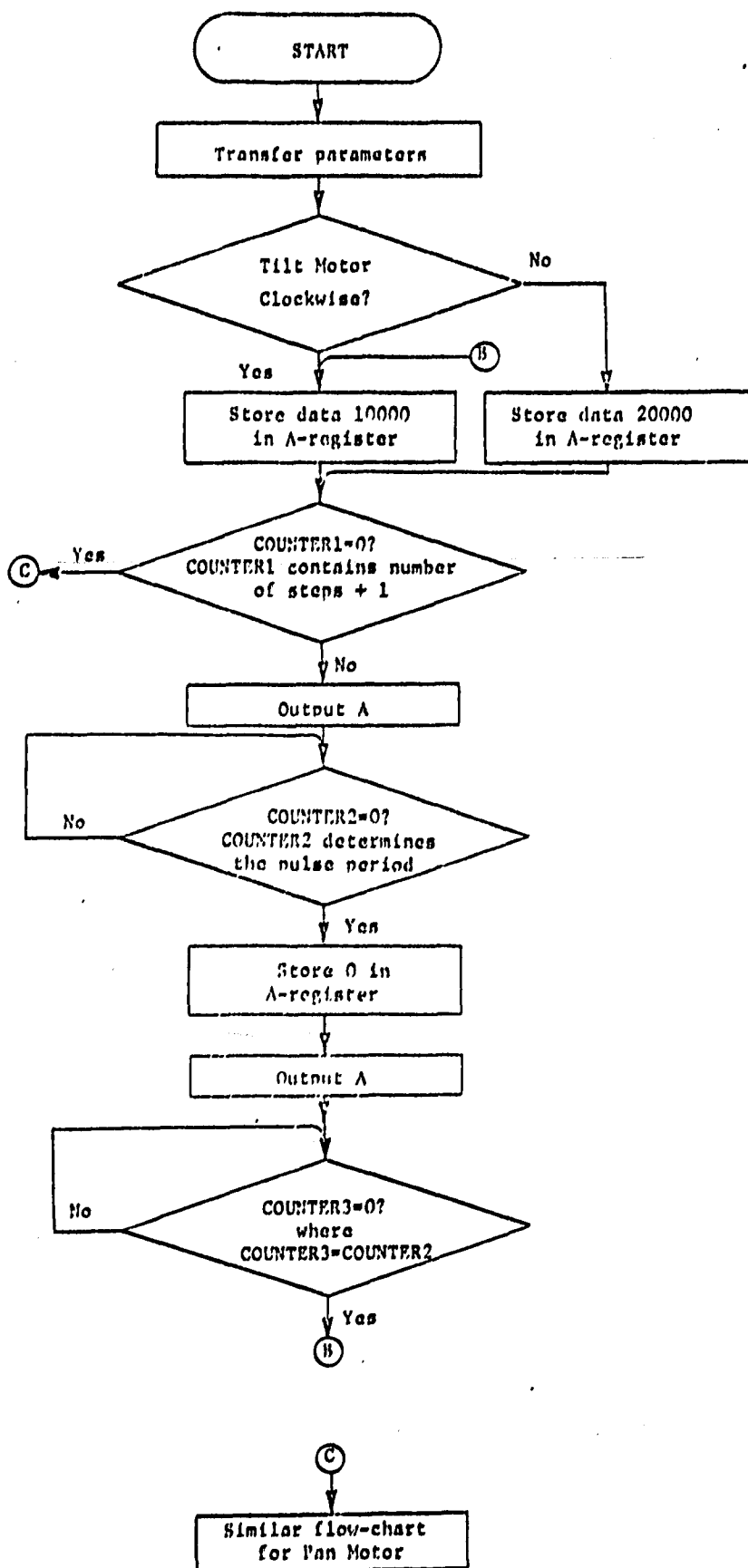


Figure 2.5 Flowchart of Pan Tilt Driver

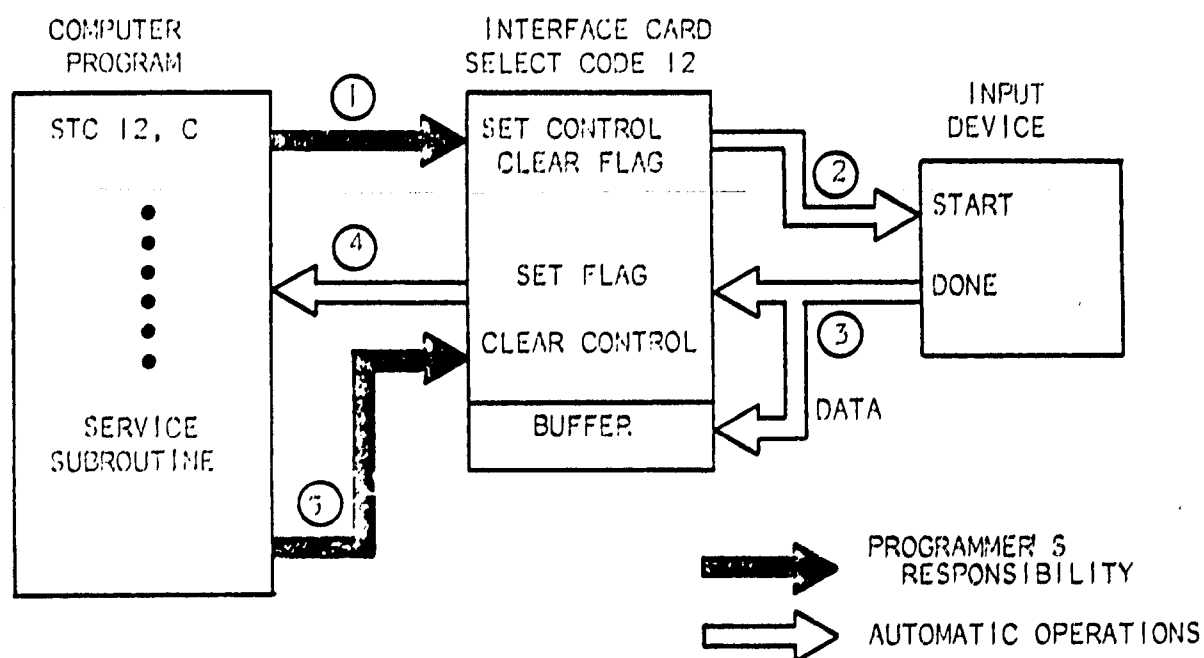
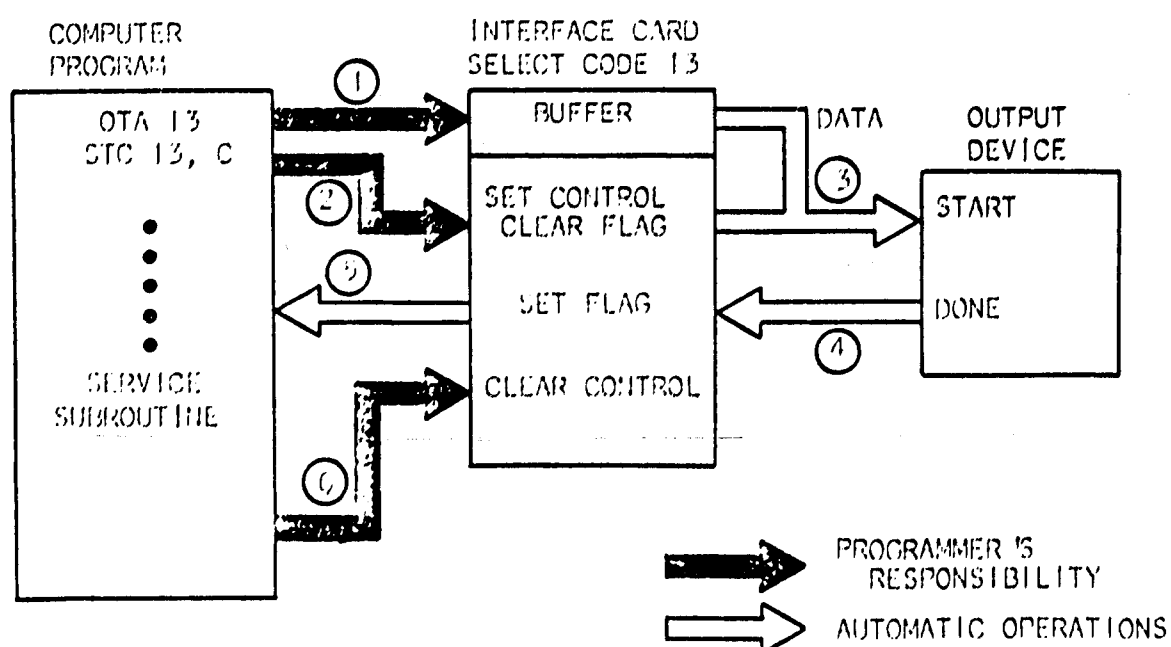
INPUT TRANSFEROUTPUT TRANSFER

Figure 2.6 Block Diagram of HP 2100 Non-interrupt Data Transfers

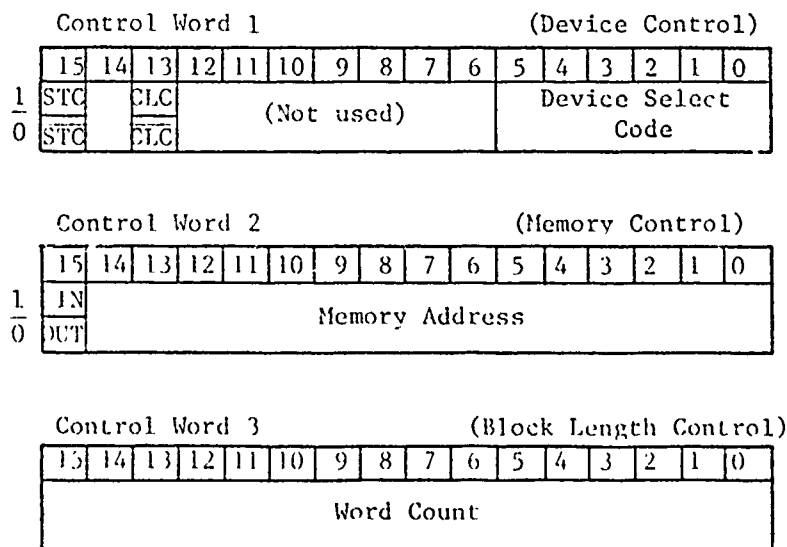
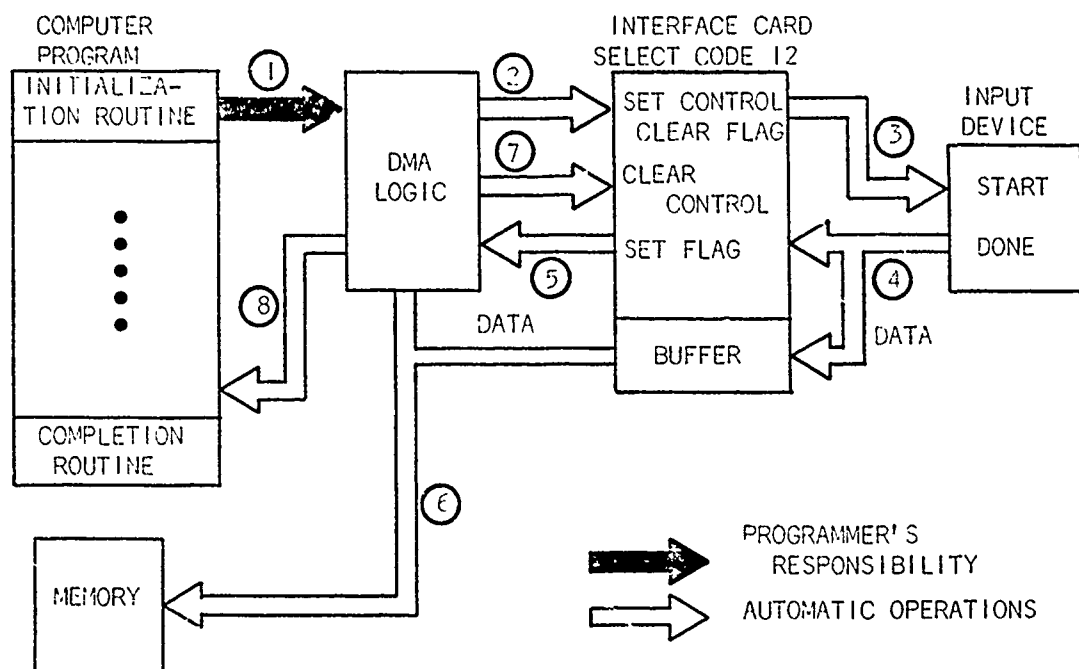


Figure 2.7 Block Diagram of HP 2100 Direct Memory Access

device. DMA accomplishes this purpose by stealing a memory cycle instead of interrupting to a service routine. When DMA is accessing memory, it has priority over the central processor's access to memory. The DMA data rate at maximum is about 1 MHz (16 bit words) but in our case it was about 500 KHz.

The DMA transfer is initiated by an initialization routine and from then on operation is under the automatic control of the hardware. The initialization routine tells the DMA hardware which direction to transfer the data, where to put the data in memory, which I/O channel to use, and how much data to transfer. This information is given by three control words. These three words must be addressed specifically to the DMA card. Figure 2.7 shows the format of the three control words. Control Word 1 identifies the I/O channel to be used, and provides two options not used in our software (Option 1 is the STC and CLF to the I/O channel at the end of each DMA cycle and Option 2 is the CLC to the I/O channel at the end of a block transfer).

Control Word 2 gives the starting memory address for the block transfer and Bit 15 of this word determines whether data is to go into memory or out of memory. Control Word 3 is the 2's complement of the number of words to be transferred into or out of memory (i.e. length of the block). This number can be from -1 to -32768 although it is obviously limited by the size of the available memory.

For the non-interrupt portion of the data transfer, four control words are sent to the camera controller: XSTART, YSTART, XSTOP, and YSTOP. The first seven bits (0 - 6) of each word identify a starting or stopping address for the software selectable camera window. The

seventh and eighth bits of each control word identifies the control word. The ninth bit of the last control word initiates the camera output.

In this manner, any portion of the camera picture (100x100) can be selected for processing. This became a necessity for the experimental system due to the limited memory available and processing speed. The software drivers for both of these processes can be found in Appendix 2.3.

2.3 Video Interface

A brief description of the camera is necessary before the interface can be discussed (2). The camera is a Reticon MC 520 photodiode camera. It provides a 100 x 100 pixel discrete analog picture. The output is pixel by pixel, row by row string of sampled-and-held pulses. Each pulse represents the grey level at a particular pixel in the image matrix. When the MC 520 is used in conjunction with its companion controller, the RS 520, (as it was in this case) various synchronizing signals are made available to the user. A pixel clock (GCLK) is synchronized with the beginning of each sampled-and-held pulse of video data and is blanked during all retrace intervals. The line enable (LEN) signal is valid during each row of 100 pixels and provides a means of knowing which row of the video data is being "clocked out" at any given time. The frame enable (FEN) signal is valid during each entire 100 x 100 pixel frame and invalid during the retrace interval between frames. It allows the user to easily synchronize his equipment with the start of any picture frame. These signals make the design of the video interface fairly

straightforward.

The interface had to meet certain requirements. First, and foremost, it had to provide some form of data reduction. Each 100 x 100 pixel image required 10,000 words of computer storage. If it were necessary that two successive image frames be stored in memory simultaneously then there would be little memory available for programs. Thus, some form of data reduction was imperative. Since the tracking algorithm utilized only those image points within the target, a means of selecting out only those particular pixels for transmission was needed. The technique employed will be described in the general description of the interface. The interface was also required to digitize the discrete analog image into a sufficient number of grey levels at speeds approaching one megahertz, the D.M.A. rate of the HP 2100.

Finally the interface was required to allow the camera to clock out image frames continuously. This was required to prevent saturation of the photodiode array. The camera operates by integrating the light incident on each photodiode for the entire period between which each photodiode is sampled. Thus allowing the camera to "sit" and only clock out frames on demand would allow it to integrate light for too long a period of time between samplings. This would result in the saturation of the photodiode array. To avoid this saturation the camera was allowed to run freely, clocking out frames continuously. When the computer signals the interface to ask for the next image frame, the interface must be able to find the beginning of the next available image frame and begin data transmission with the first

pixel in the frame. The means by which this requirement and all the other requirements were satisfied will become clear as the interface is described.

The camera-computer interface consists of two major subsystems. The digital subsystem allows the user to select a desired portion of an image for transmission to the computer. The analog subsystem performs the analog to digital conversions necessary for communication with a digital computer. The entire interface resides on one 4.5 by 9 inch wire wrap circuit card. It is installed in the RS 520 controller which has extra slots in its card cage meant specifically for user hardware such as the interface.

The digital subsystem allows the user to select from software a rectangular (or square) "window" or subsection of the entire image for transmission to the computer. The user specifies the parameters of the desired window by transmitting, to the interface, four control codes specified in software. A frame is acquired by sending "go" command which is usually issued with the last control code.

The operation of the digital subsystem is best understood by separating its operation into an initialization mode and a run mode. Figures 2.8, 2.9, and 2.10 should be referred to throughout this description. The initialization mode allows the user to program the interface which specifies the software selectable window. Bits seven and eight of the 16 bit computer word are used to select which of four eight bit (only seven bits used) latches is to be initialized. Bits zero to six of each word are then used to initialize each of the four latches. Each latch contains the coordinates within the 100

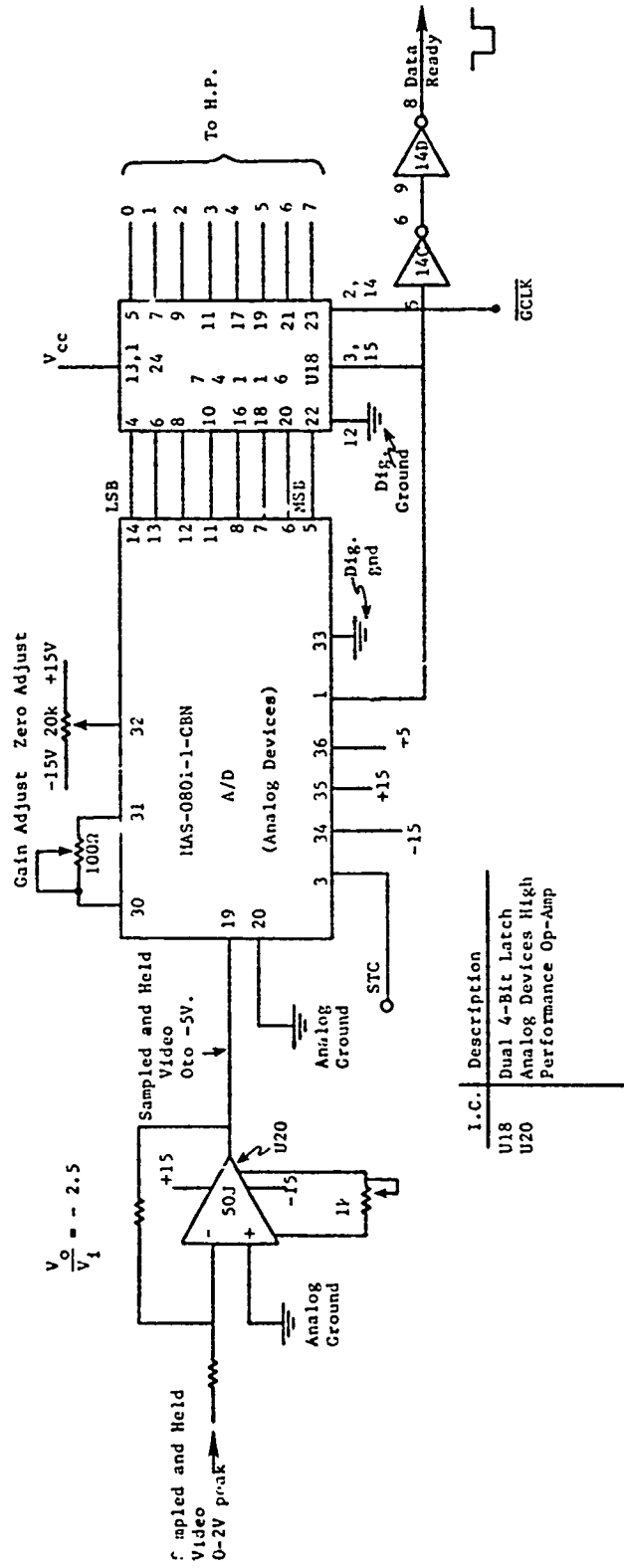


Figure 2.9 Analog Subsection of Video Interface

VIDEO INTERFACE (Digital Subsystem)

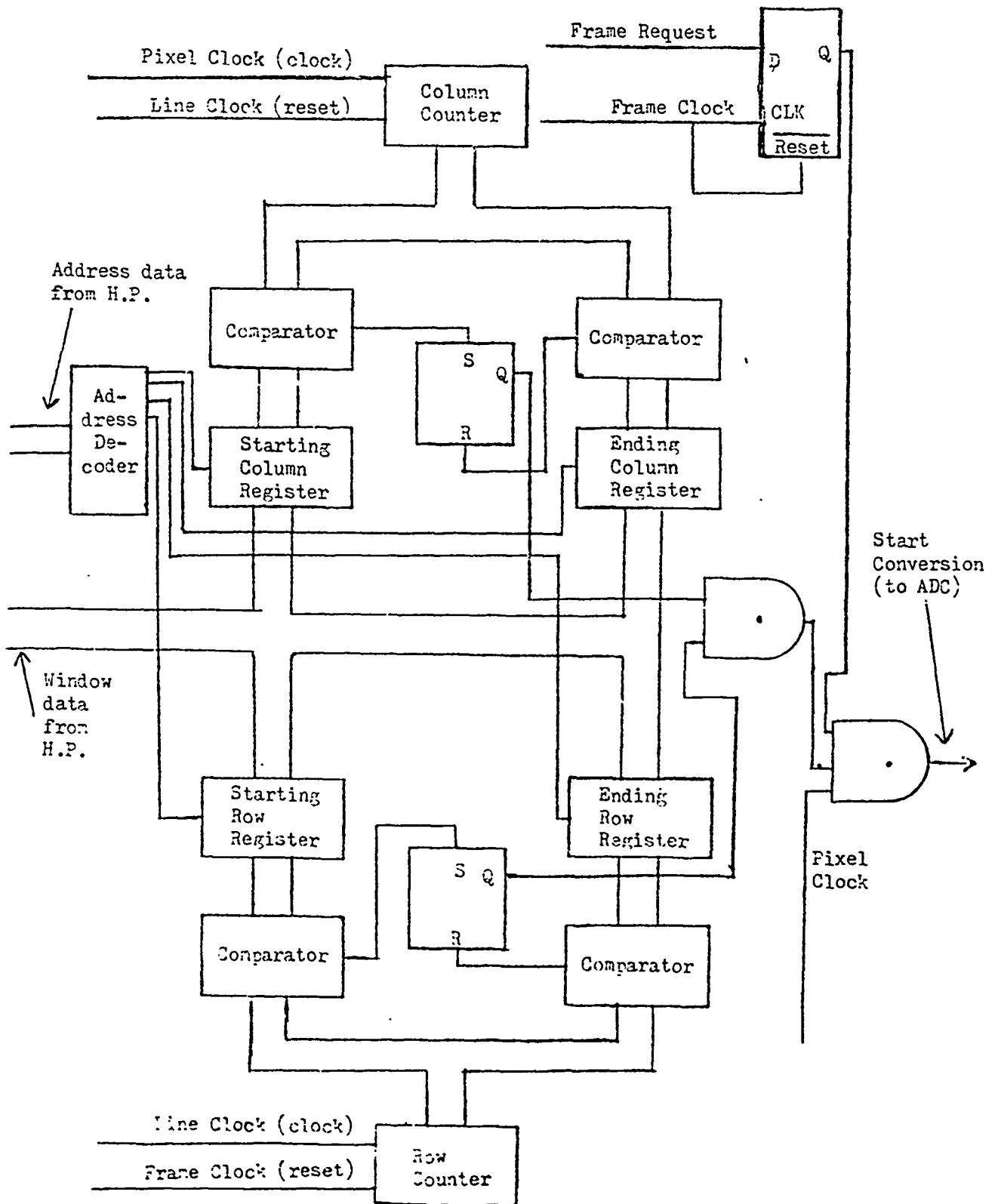


Figure 2.10 Block Diagram of Digital Subsection of Video Interface

x 100 pixel array of the start or stop position of the column or row desired. The four codes are referred to as XSTART, XSTOP, YSTART, and YSTOP and indicate the starting column, ending column, starting row, and ending row, respectively of the desired window. Figures 2.8 and 2.10 provide an illustration of this.

The term "run mode" is actually a slight misnomer since the digital subsystem is actually free running while its output is disabled until a "go" command is issued. The four 4 bit counters (U1, U2, U11, and U12) are used in pairs to form two 8 bit counters in order to keep track of the pixel whose value is being clocked out. The "X-counter" is clocked by the logical complement of GCLK, the pixel clock, which occurs as each successive pixel clocked out. It is reset by the LEN signal. Thus the value of the X-counter at any given time is the column coordinate of the pixel being clocked out at that time. The "Y-counter", which is clocked by the LEN signal and reset by the FEN signal, keeps track in a similar fashion of the row coordinate of the pixel being clocked out. Thus the values of the X and Y counters taken together provide the coordinates of the pixel being clocked out at any given time.

Digital comparators (U3, U4, U9, and U10) are used to compare the location of the pixel being clocked out at a particular time to the location of the window limits as stored in the four latches during initialization. The output of the comparators provides the information necessary to know whether the pixel being clocked out is within the specified window. The XSTART and XSTOP comparators set and reset, respectively a flip-flop (U16A) whose output status tells whether

the column coordinate of the pixel being clocked out is within the specified window. The YSTART and YSTOP comparators provide the same information about the row coordinate by setting and resetting another similar flip-flop (U16B). Thus, by combining the outputs of the two flip-flops using a logical AND function provides a single digital signal whose status tells whether the pixel being clocked out is within the desired window. The window status signal combined with information as to whether a frame request has been made is used to appropriately enable or disable the clock to the analog subsystem.

The user makes a request to acquire a frame by setting bit nine of the last control word. This bit is clocked into a D-type flip-flop (U17B) by the FEN signal. This causes the output of the flip-flop to be synchronized with the start of the next available frame. The output of this flip-flop combined by a logical AND function (U15) with the window status signal is used to enable the clock to the analog subsystem.

The main task of the analog subsystem is to perform analog-to-digital conversion. This subsystem consists of high performance operational amplifier, and eight bit one megahertz analog-to-digital converter, and an eight bit latch. A schematic diagram is shown in Figure 2.9. The operational amplifier gain of -2.5 serves to scale the 2 volt maximum sampled-and-held video from the camera to a 0 to -5 volt range so as to make use of the entire dynamic range of the ADC. The ADC is capable of performing conversions at speeds up to one megahertz. The actual clock rate is set by adjusting the internal clock of the RS 520 camera controller. The latch connected to the ADC output holds

each conversion result for an entire conversion period to allow sufficient time for the computer to read each piece of data. The start-conversion (STC) signal to the ADC is the output of the digital subsystem. Thus, the ADC only receives STC signals when a frame has been requested by the user and the pixel being clocked out is within the user specified window.

The operational details of the control codes required to initialize the interface are as follows:

CODE	BIT 9	8	7	6	5	4	3	2	1	0
XSTART	0	1	0	X	X	X	X	X	X	X
XSTOP	0	0	0	X	X	X	X	X	X	X
YSTART	0	0	1	X	X	X	X	X	X	X
YSTOP	0	1	1	X	X	X	X	X	X	X
YSTOP with "go"	1	1	1	X	X	X	X	X	X	X

where X is user specified.

One important point to note is that after a frame request is made by sending the last control code with bit 9 set, this bit must be reset and then set again when a frame is next requested. Failure to reset bit 9 of the latched output port between frame requests will render the frame synchronizing apparatus ineffective.

The interface is an open loop device. This means that it provides no handshaking lines (other than the usual strobe or flag) either when latching control codes or transmitting data to the computer. To assure consistent operation, the system clock (internal to RS 250)

must be set at less than 800 KHz. In this way the D.M.A. rate of the HP 2100 is not exceeded and the propagation delay inherent in latching the output of the ADC is taken into account.

2.4 Tracking

The purpose of this section of the system is to estimate the affine parameters using the TSVIP algorithm.[5]. A brief development of the algorithm is given below (see ONR-CR-233-092-1 for complete details):

$$\underline{d}_c = D_c \underline{a}_c \quad (2.1)$$

where

$\underline{d}_c$ is the $N \times 1$ scene difference vector
 $\underline{a}_c$ is the 4×1 total affine parameter vector
 D_c is the constrained $N \times 4$ matrix

D_c is

$$D_c = [P_c : G] \quad (2.2)$$

where

P_c is the $N \times 2$ matrix of weighted spatial derivatives
 G is the $N \times 2$ matrix of spatial derivatives

$$D_c^+ = [P_c : G]^+ = \begin{bmatrix} P_c^+ & C^+ P_c^+ G C^+ \end{bmatrix} \quad (2.3)$$

where

$$C^+ = [I - P_c P_c^+] G \quad (2.4)$$

The affine vector can be estimated by

$$\underline{a}_c = \begin{bmatrix} \cdot \\ \underline{a}_{cC} \\ \cdot \\ \underline{b}_{cC} \end{bmatrix} \quad (2.5)$$

where

$$\begin{aligned} \underline{a}_{cC} &= (P_c^+ - P_c^+ G C^+) \underline{d}_c \\ \underline{b}_{cC} &= C^+ \underline{d}_c \end{aligned} \quad (2.6)$$

Software was developed to implement this algorithm with two intentions in mind: first, ease of implementation; and second, ease of modification. Therefore the software took the form of one overall control algorithm and many array processing subroutines. The software is well documented and fairly straightforward following the development found on the previous page. The software can be found in Appendix 2.3.

2.5 Open Loop Tracking

Upon completion and testing of the software and hardware, the relationship between relative position and estimated position was examined. Figures 2.11, 2.12, and 2.13 demonstrate the sensitivity of the experimental system to noise. A great deal of noise smoothing had to be incorporated into the tracking system in order to compensate for the slow sampling rate (once every 30 seconds) and a very noisy lighting environment. With a greater processing speed (such as CCD technology is now capable of), a faster sampling rate can be achieved which theoretically should decrease the magnitude of changes in the target's lighting environment.

After the effect of noise has been reduced, Figure 2.13 shows that the TSVIP algorithm does generate linear position estimates for small target movements from the origin. With a fast enough system sampling rate, all target movements can be limited to this range.

2.6 Closed Loop Tracking

Once the TSVIP algorithm produces positional estimates, it is necessary to translate them into pan tilt control signals. Since a priori knowledge of the target size and distance to the sensor is

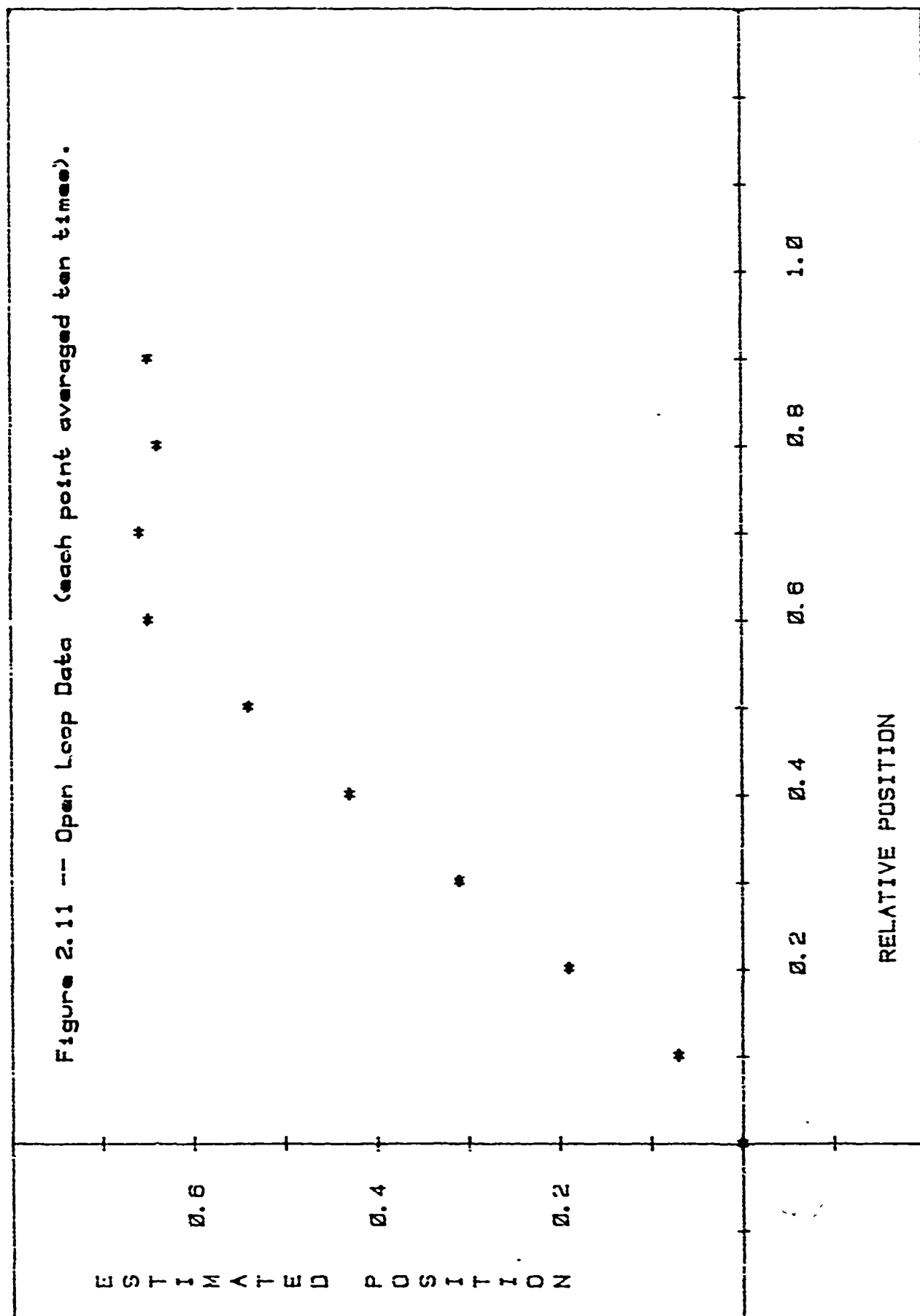
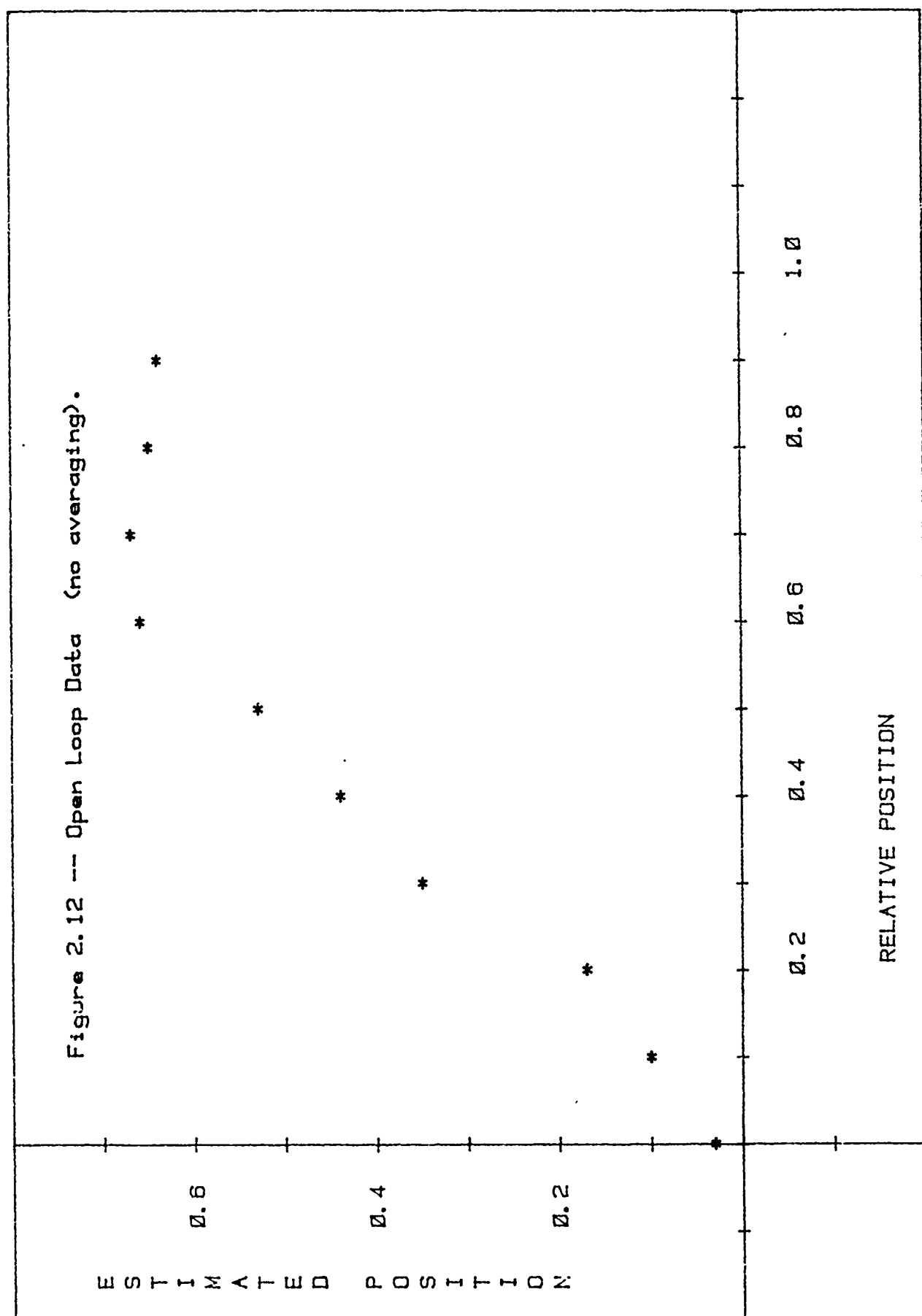
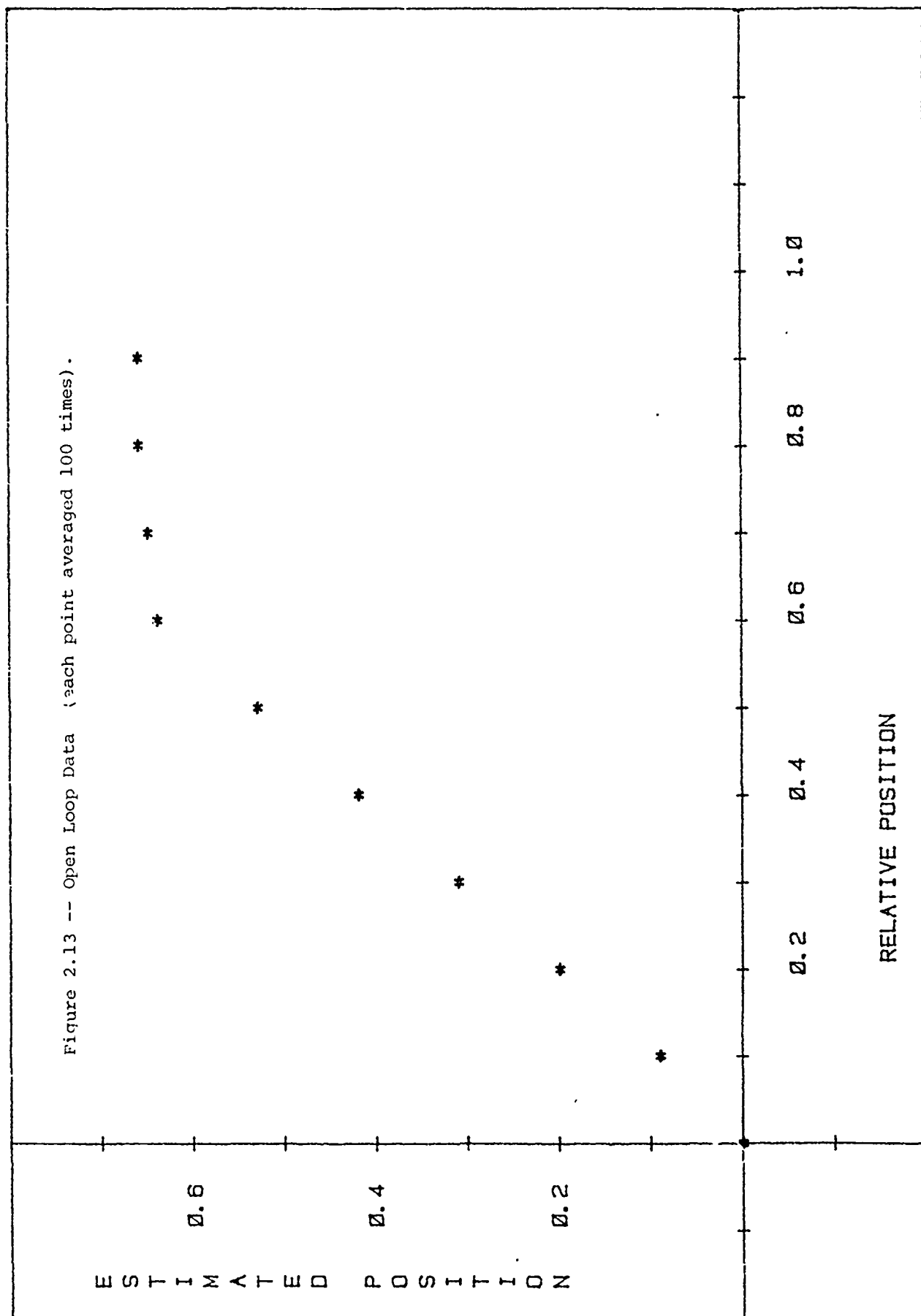


Figure 2.12 -- Open Loop Data (no averaging).





not available, an alternate method is needed. A straightforward method is to move the camera in the direction followed by the target with a constant step size until the error is within certain limits. This method generally guaranteed convergence at the expense of processing time.

A second approach has been tested where the values for guidance control signals were found by interpolating between the previous and present position estimates and the previous control signal. This algorithm was much faster than the previous one but was very sensitive to noisy position estimates as long as the target was in the linear region of the TSVIP algorithm. Outside of this region, the response time of this algorithm was decreased although it was still better than that of the first method. This deficiency could be avoided provided that the system sampling rate was fast enough to limit target motion to this range.

A real environment tracking sequence of scenes copied from the graphics display can be found in Appendix 2.4. The initial scene shows the position of the target after it has moved away from the origin. The remaining images display the target position after computer simulating the TSVIP algorithm attempts to center the target in its coordinate system. Note that the sequence ends when the computer has successfully tracked the target back to its origin.

Closing the feedback loop allows any errors generated from the previous position estimates or camera position control signals to be corrected as can be seen in Appendix 2.4. It took eight iterations for our tracking system to track the target back to its origin but

with an increase in processor speed as with CCD technology real time video tracking becomes a reality. Since it is impossible to increase the computation speed of the HP 2100, it was necessary to slow down the target motion. Moving at a speed which kept the target in the field of view of the camera, any single target which could be segmented from the background was capable of being tracked for any length of time as long as the above conditions were met.

CHAPTER III

CCD IMPLEMENTATION

3.1 Introduction

Algorithms such as the TSVIP require the manipulation of matrices with variable elements. Matrix multiplication is the most commonly performed operation, but in some cases matrix inversion is also necessary. The TSVIP algorithm (in reduced form) requires the inversion of a 2×2 matrix with variable elements. This inversion can be performed using only one analog divider (in addition to CCD registers and analog multipliers). The block diagram and schematic diagrams for the implementation of the complete TSVIP were presented in the final report for the period August 1978 - August 1979 (Report ONR-CR233-092-1). The processor requires 16 subsystems, all of which, except the spatial derivative estimator use CCD devices. Some require analog multipliers also, and only one, as mentioned above, an analog divider. It is suggested in [6] that the use of analog dividers can be avoided in matrix inversion if the method of inversion by products is used. However this still requires the use of analog multipliers plus n digital divisions for the inversion of an $n \times n$ matrix. For a 2×2 matrix it seems that the disadvantages of this method outweigh the possible advantages, and the use of a single analog divider is adequate.

Several methods for the implementation of programmable filters which can be extended to matrix operations have been reported in the literature. The most promising one, [7], uses a combination of CCDs and digital shift registers for the multiplication of a variable vector by a matrix with elements known a priori. More details on this approach will be given in Sect. 3.3. The method could be used also for a variable matrix, but the speed of operation would

then be slowed down by the loading time of the shift register, plus the A/D conversion time. Except for this possibility, no other method for variable matrix operations has been reported, to our knowledge, which avoids the use of analog multipliers.

3.2 Sum of Products (SOP) Matrix Manipulator

Carroll, [6], uses only serial input/parallel output (SIPO) devices to implement his basic SOP cell shown in Fig. 3.1. It should be noted that although apparently simple, this cell has the disadvantage of producing the elements of the resulting matrix at different time instants. Consequently additional circuitry not shown in the figure becomes necessary in order to store the elements of the resultant matrix for use in other parts of the circuit, or as a final output. To obtain the product of two matrices, one matrix is first loaded by columns and then the second matrix is "paraded" left-to-right by rows, at consecutive clock pulses. The product of an $m \times p$ matrix A by a $p \times n$ matrix B can be accomplished with p SOP cells in $2m+n-1$ clock pulses. This includes the loading of A and the "parading" of B. With reference to Fig. 3.1, if we want to multiply two 2×2 matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \quad (3.2.1)$$

A is loaded during the first two clock pulses. During the 3rd clock pulse b_{11} is loaded at input 1 and b_{21} at input 2. The output at t_3 is then c_{11} at output 1, where $c_{11} = a_{11}b_{11} + a_{12}b_{21}$, and zero at output 2. Table 3.1 shows the outputs from t_1 to t_5 . Note that five clock pulses are needed, in agreement with

$$\begin{aligned} T &= 2m+n-1 \\ &= 4+2-1 \\ &= 5 \end{aligned} \quad (3.2.2)$$

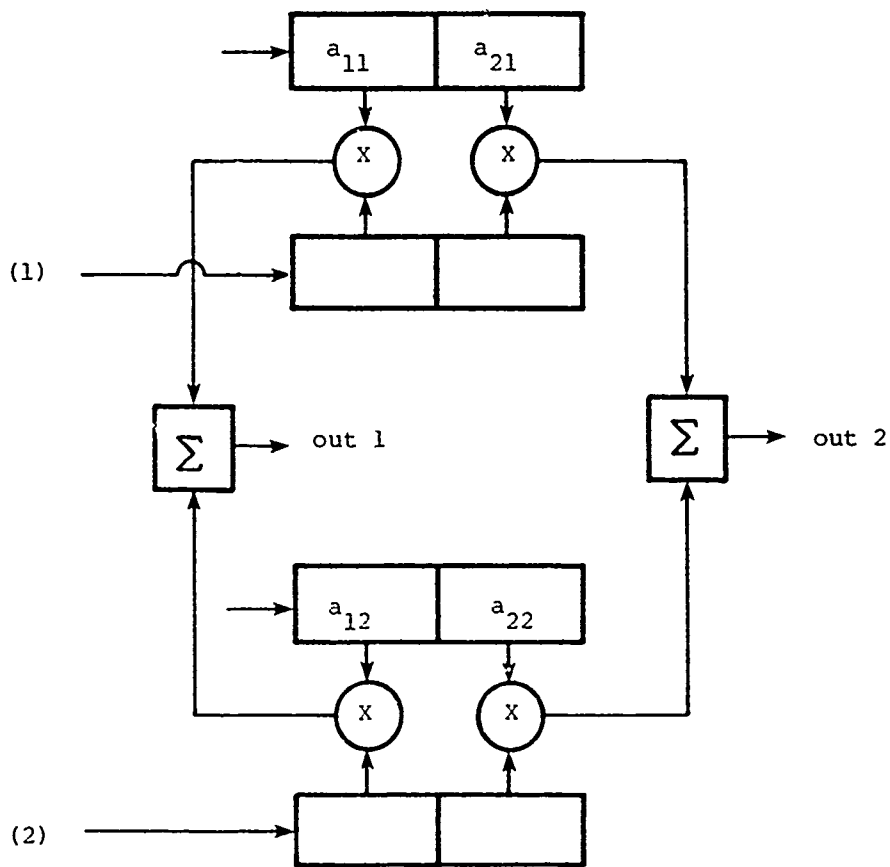


Fig. 3.1 Basic SOP Cell

- (1) Load row 1 of matrix [B] starting with b_{11} at clock pulse 1
- (2) Load row 2 of matrix [B] starting with b_{21} at clock pulse 1

Exactly the same cell can be used to multiply a $q \times 2$ matrix by a $2 \times n$ matrix. The required time will then be

$$T = 2q + n - 1 \text{ clock pulses} \quad (3.2.3)$$

TABLE 3.1

	t_1	t_2	t_3	t_4	t_5
OUT 1	0	0	c_{11}	c_{12}	0
OUT 2	0	0	0	c_{21}	c_{22}
					
	loading A				

With reference to the block for the implementation of

$$[P]^+ = [P_c^T \ P_c]^{-1} [P_c]^T \quad (3.2.4)$$

given in the previous report and repeated in Fig. 3.2 for convenience, we see that it can be implemented by the circuit of Fig. 3.1, because $[P_c^T P_c]^{-1}$ is 2×2 and P_c^T is $2 \times N$. For the subsystem of Fig. 3.2, we had

$$[P_c] = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ . & . \\ . & . \\ p_{N1} & p_{N2} \end{bmatrix} \quad (3.2.5)$$

$$[P_c]^T [P_c]^{-1} \triangleq \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \triangleq [q] \quad (3.2.6)$$

$$[P]^+ = [[P_c]^T [P_c]]^{-1} [P_c]^T \triangleq \begin{bmatrix} q_{11} & q_{21} & \cdots & q_{N1} \\ q_{12} & q_{22} & \cdots & q_{N2} \end{bmatrix} \quad (3.2.7)$$

If the system of Fig. 3.1 is used, the operation would require $N+3$ clock pulses (at frequency NF). Using the circuit of Fig. 3.2 only N clock pulses are required. The CCD devices of Fig. 3.2 are more complex than those of Fig. 3.1, however, but the outputs are all obtained simultaneously by means of analog switches at the parallel outputs of the SIPO devices. SIPO devices can be operated in general at a higher frequency than PISO devices, and if a very high frequency of operation is required, the approach of Fig. 3.1 should be considered.

3.3 Digital/Analog Matrix Manipulator

3.3.1 A Programmable Digital/Analog Correlator

Some of the programmable transversal filters or correlators reported in the literature use digital storage and a simple FET analog multiplier, [3]. As already noted in the previous report, the usefulness of such devices is doubtful. Recently, [8], a different architecture which utilizes digital storage and multiplying DACs has been reported although the most practical approach seems to be the one reported in reference [7]. A programmable digital/analog correlator capable of performing an n -stage programmable correlation which does not require multiplication is shown in Fig. 3.3. The output of this device in the z -domain is

$$\begin{aligned} \frac{V_{\text{out}}(z)}{V_{\text{in}}(z)} &= H(z) \\ &= \sum_{n=0}^{N-1} h_n z^{-n} \end{aligned} \quad (3.3.1.1)$$

where

$$h_n = \sum_{k=0}^{M-1} h_n^k 2^{-k} \quad (3.3.1.2)$$

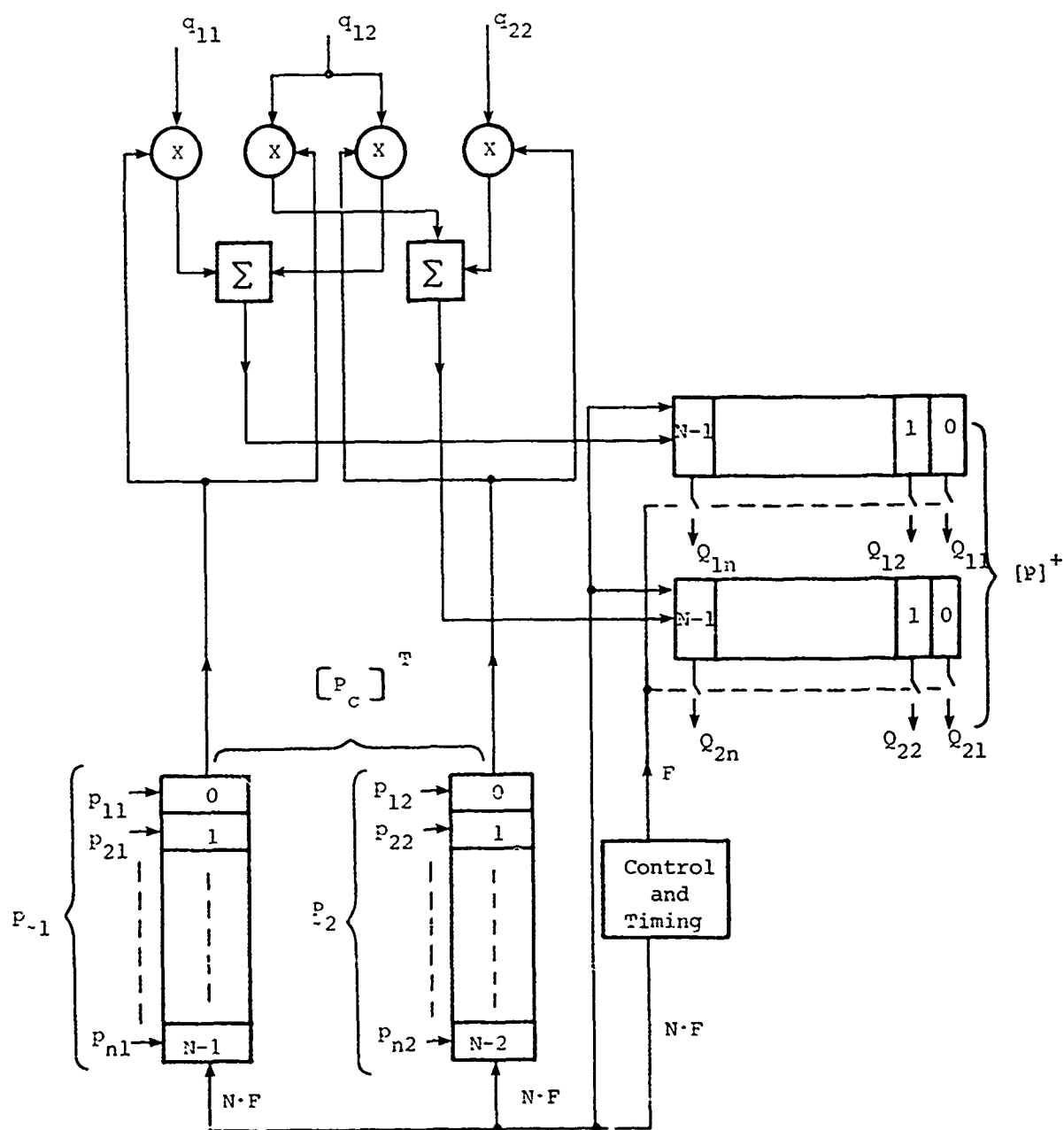


Figure 3.2 The implementation of $[P_c]^+$ using CCDs and Analog multiplier

is the n^{th} weight, represented with M bit precision. Notice that M delay lines and M digital shift registers are needed. The analog signal at each CCD tap output is "multiplied" by the binary value h_n^k (which is either 0 or 1) by either sensing or not sensing each analog charge packet once the n values of V_{in} have been loaded into the CCD device. The 2^{-k} attenuation ($0 \leq k \leq M-1$) at the input of each delay line is achieved by means of capacitive ratio techniques. An off-chip integrating amplifier is used to sum the correlator outputs. The slew rate of this amplifier limited the sampling rate of the correlator reported in [7] to 500 KHz.

3.3.2 Use of the Programmable Digital/Analog Correlator for Matrix Operations

Some of the algorithms described in Chapter V of this report require the computation of image moments of the type

$$M_x = \sum_{i=-m}^m \sum_{j=-n}^n p'(i,j) x_i \quad (3.3.2.1)$$

where the windowed image $p'(i,j)$ is of size $(2m+1)(2n+1)$ and x_i is the distance in the x direction from the center of the window. The x_s can then be considered as weights of the image intensities $p'(i,j)$. Formula (3.3.2.1) can be rewritten as

$$M_x = \sum_{i=-m}^m p'(i,-n) x_i = \sum_{i=-m}^m p'(i,(-n+1)) x_i + \dots + \sum_{i=-m}^m p'(i,n) x_i \quad (3.3.2.2)$$

Each one of the addends in (3.3.2.2) can be obtained with a correlator of the type shown in Fig. 3.3. From (3.3.2.2),

$$M_k \triangleq \sum_{i=-m}^m p'(i,k) x_i, \quad -n \leq k \leq n \quad (3.3.2.3)$$

$$M_x = \sum_{k=-n}^n M_k \quad (3.3.2.4)$$

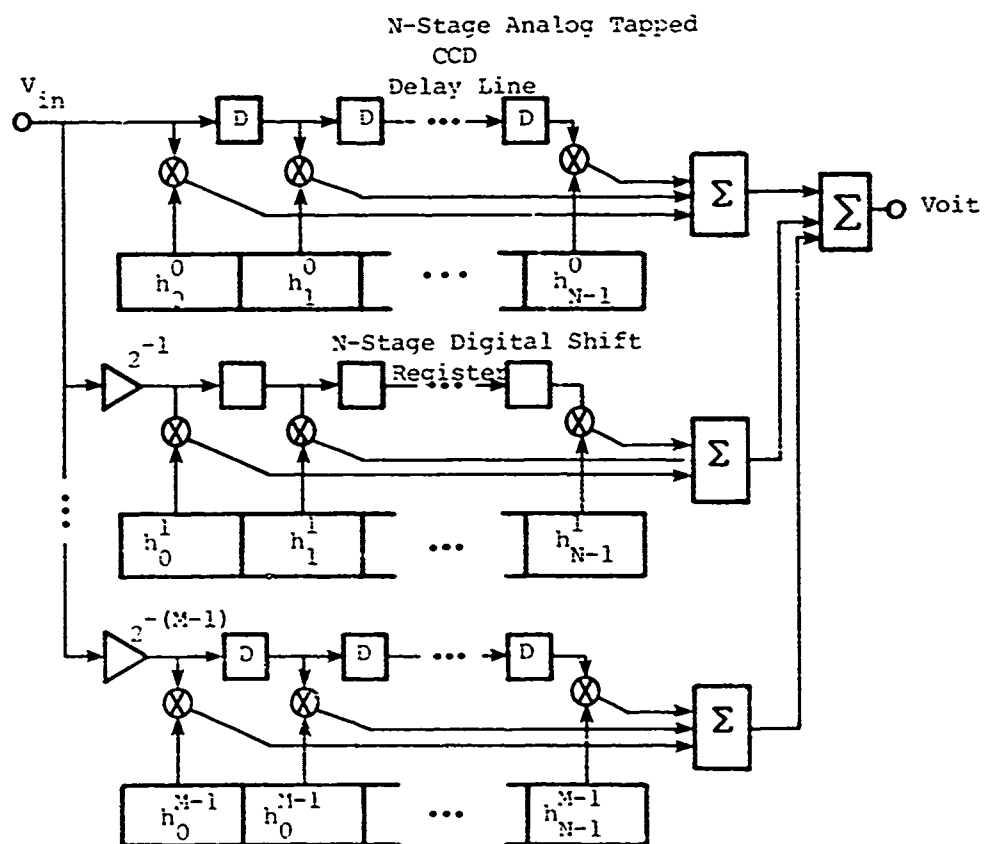


Figure 3.3 A programmable digital/analog correlator

The correlator to implement (3.3.2.3) requires $2m+1$ taps for the delay lines and $2m+1$ bits for the digital shift registers. The number M of delay lines and shift registers is determined by the required accuracy. A total of $2n+1$ correlators is necessary to implement M_x . The final configuration is as shown in Fig. 3.4, where every block has the architecture of Fig. 3.3. Notice that the $p'(i,j)$ are loaded serially into the CCD devices and that x_i , a value known as priori, is loaded only once into the digital shift registers.

The operation performed by the system of Fig. 3.4 is similar to the multiplication of a matrix by a vector. In this case the vector $\underline{x}$ being a constant vector. If it is desired to perform a similar multiplication with a vector of variable elements, an ADC is necessary in order to obtain the digital representation of the vector elements. In addition, each digital shift register would have to be reloaded for each new multiplication. Since the rows of the matrix also have to be reloaded serially into the CCD devices, the processing time would only be slowed down by the A to D conversion time.

3.3.3 Use of the Digital/Analog Correlator for the Implementation of the TSVIP

With reference again to the implementation of the pseudoinverse matrix $[P_C]^+$ in the TSVIP, we have, from (3.2.5) and 3.2.6),

$$\begin{aligned}
 [P_C]^+ &= \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{N1} \\ p_{12} & p_{22} & \cdots & p_{N2} \end{bmatrix} \\
 &= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \end{bmatrix} \quad (3.3.3.1)
 \end{aligned}$$

where

$$\begin{aligned}
 c_{11} &= q_{11}p_{11} + p_{12}p_{12} \\
 c_{12} &= q_{11}p_{21} + q_{12}p_{22} \\
 c_{21} &= q_{12}p_{11} + q_{22}p_{12} \quad , \text{ etc.}
 \end{aligned}$$

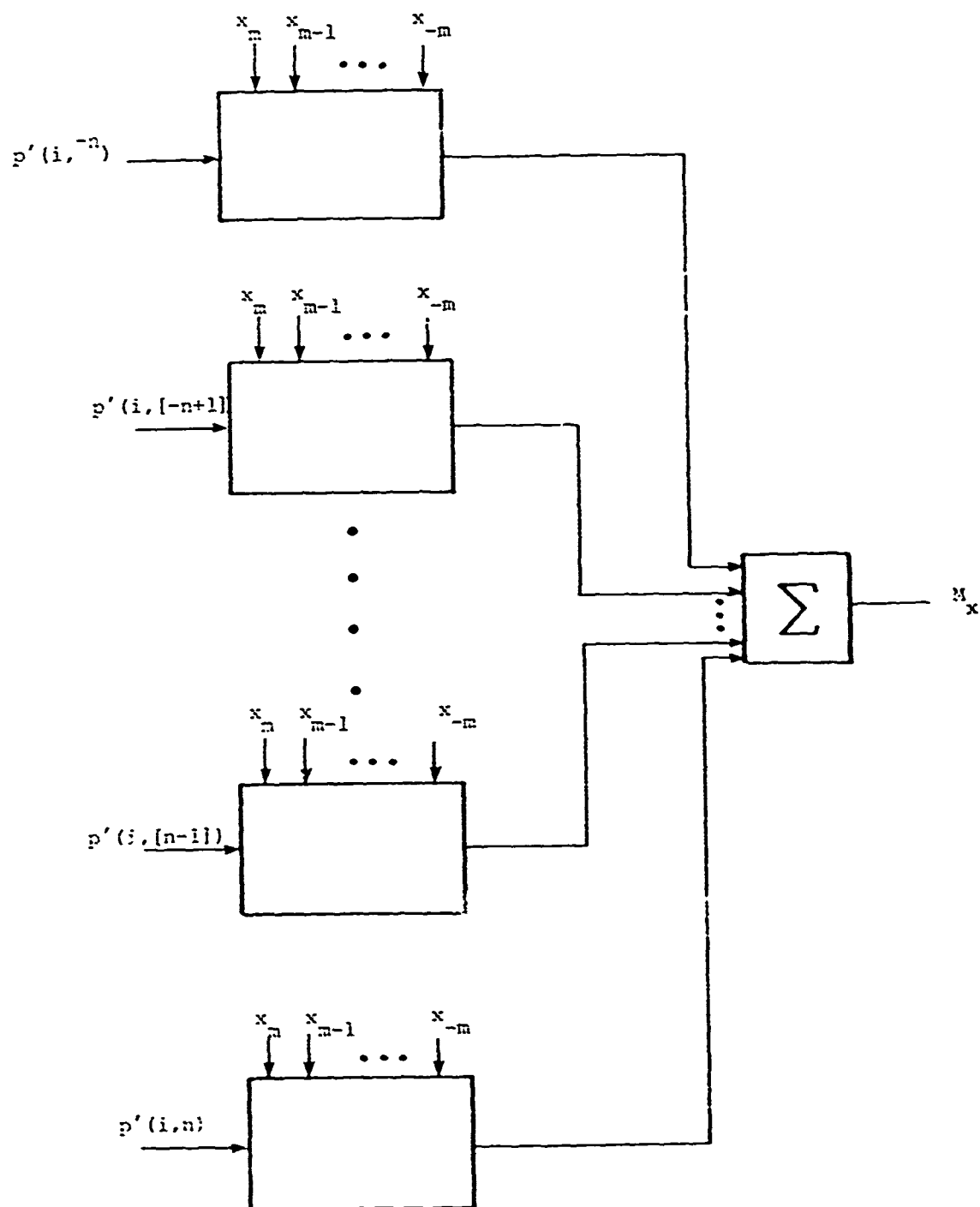


Figure 3.4 System for computing: $y_x = \sum_{i=-m}^m \sum_{j=-n}^n p'(i, j) x_i$

Defining

$$p_{11} = \sum_{k=0}^{M-1} h_{11}^k 2^{-k} \quad (3.3.3.2a)$$

$$p_{12} = \sum_{k=0}^{M-1} h_{12}^k 2^{-k} \quad (3.3.3.2b)$$

$$p_{jm} = \sum_{k=0}^{M-1} h_{jm}^k 2^{-k} \quad j = 1, 2 ; m = 1, 2, \dots, N \quad (3.3.3.2c)$$

the matrix multiplication (3.3.3.1) can be expressed in terms of the multiplication of the 2×2 matrix $[q]$ and the vectors

$$p_j \triangleq \begin{bmatrix} p_{j1} \\ p_{j2} \end{bmatrix}, \quad j = 1, 2, \dots, N \quad (3.3.3.3)$$

as follows,

$$c_j \triangleq \begin{bmatrix} c_{1j} \\ c_{2j} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} p_{j1} \\ p_{j2} \end{bmatrix}, \quad j = 1, 2, \dots, N \quad (3.3.3.4)$$

Hence,

$$[p_c]^+ = [c_1 \ c_2 \ c_3 \ \dots \ c_N] \quad (3.3.3.5)$$

The element c_{11} of c_1 can be implemented as shown in Fig. 3.5. The element c_{12} of c_1 is implemented by exactly the same architecture with the input to the delay line changed to q_{12} , q_{22} instead of q_{11} , q_{12} . For both elements, the input to the ADC is p_{11} , p_{12} , sequentially, which indicates that only one ADC is needed for each c_1 . Notice that the timing must be such that when q_{11} is the output of the delay and q_{12} is at the input to the delay, p_{11} must have been converted and shifted to the right, and p_{12} must have been converted, as indicated in Fig. 3.5. Notice also that the hardware could be reduced by half, since q_{11} , q_{12} and q_{22} could be entered sequentially (here the symmetry of the matrix allows us to use three inputs, instead of four) to the delay, producing c_{11} and c_{12} sequentially at the output. The processing time would, however,

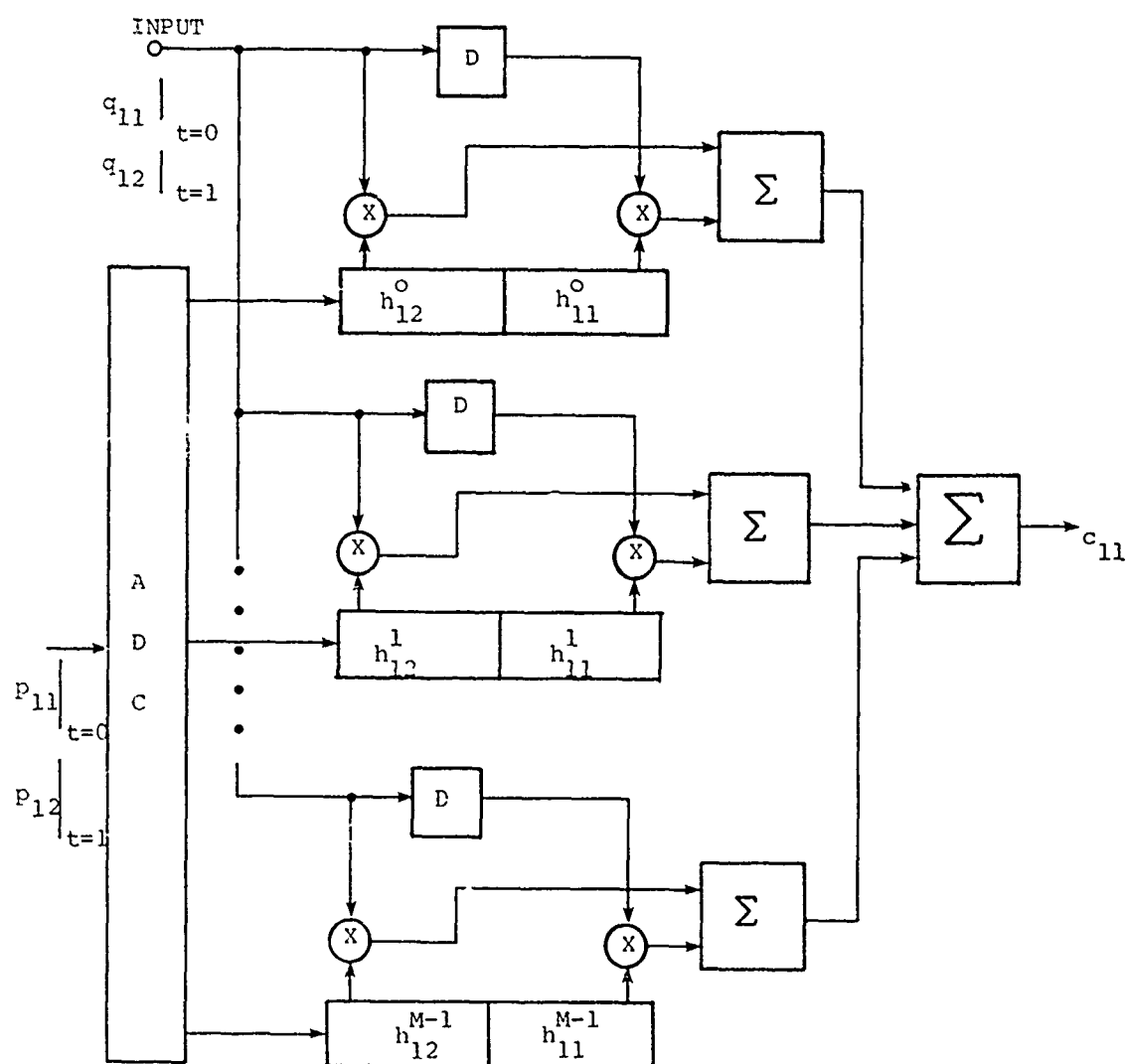


Figure 3.5 Digital/analog implementation of element c_{11} of matrix $[p_c]^+$

be larger, and means of storing c_{11} while c_{12} is being computed would be necessary.

If the parallel approach is used, $2N$ blocks are needed. If the sequential approach is used, N blocks are needed, plus the extra circuitry necessary to store c_{1n} while c_{2n} is being computed. The all parallel block diagram is shown in Fig. 3.6.

Another possibility is the all-serial processor, in which a single block of the type shown in Fig. 3.5 is used. The p_j s are converted and loaded sequentially into the digital shift register, from $j=1$ to $j=N$. For each p_j , q_{11} , q_{12} and q_{22} are loaded sequentially into the CCD delay line, and the multiplication performed. The elements of the matrix $[P_c]^+$, c_{jk} , appear sequentially at the output of the device and must be stored. The processing time will be N times larger than that for the parallel-serial processor, but the hardware will be reduced from N to 1. Notice that only one ADC is necessary in this case. The times required for the three cases are:

a) All parallel processor,

$$T_p = (\text{ADC})_{\text{time}} + 2t \quad (3.3.3.6)$$

where t is a clock pulse time.

b) Serial-parallel processor,

$$T_{sp} = (\text{ADC})_{\text{time}} + 3t \quad (3.3.3.7)$$

c) All-serial processor,

$$\begin{aligned} T_s &= N \cdot (\text{ADC})_{\text{time}} + N \cdot 3t \quad (3.3.3.8) \\ &= N \cdot T_{sp} \end{aligned}$$

From the point of view of speed, the serial-parallel processor represents the best approach, because it requires considerably less hardware than the all-parallel, with a very small increase in processing time.

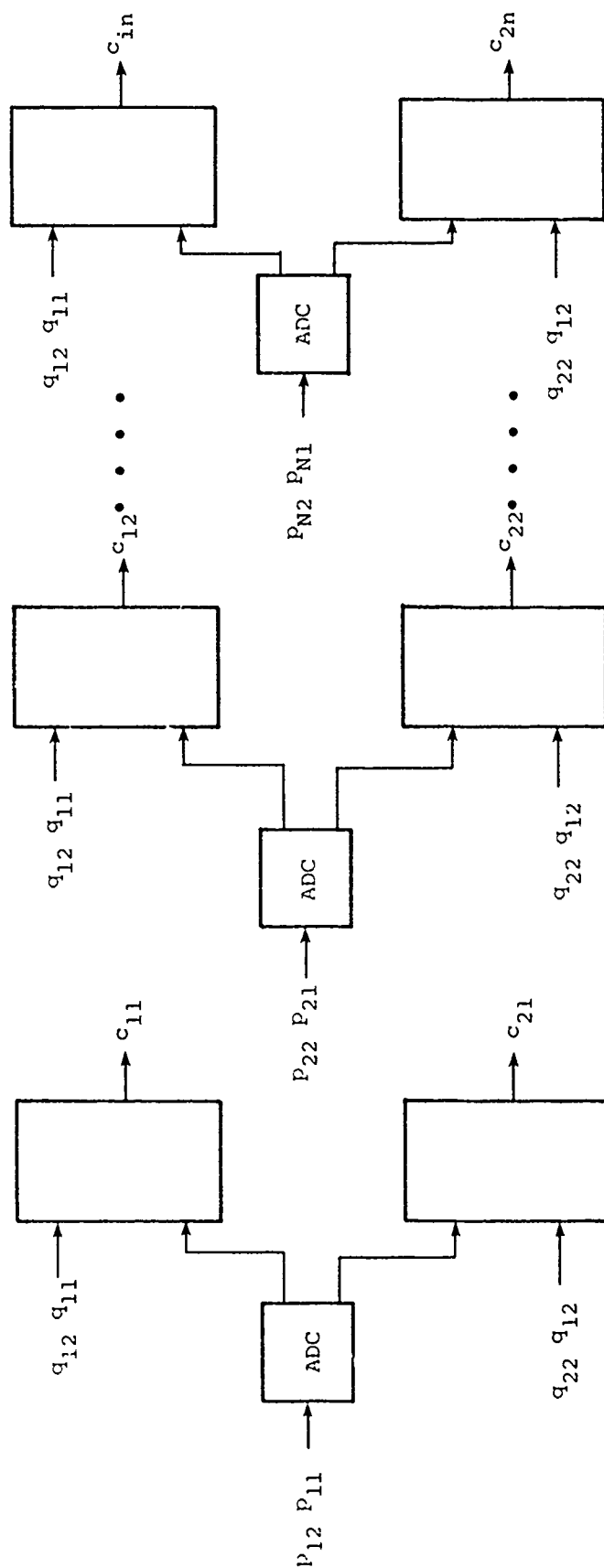


Figure 3.6 All parallel digital/analogy processor (timing not indicated)

For the all-analog system of Fig. 3.2, N clock pulses at a frequency F are needed for one computation. In that case it was assumed that the PISO devices could operate at a frequency NF . If the same assumption is made for the digital/analog system, the all serial processor would require a time equal to 3 clock pulses of the basic frequency F plus N times the ADC time. High speed ADCs have conversion rates of about 100 ns for 8 bit accuracy. For a basic frequency of 1 MHz and $N = 9$, the all serial processor could compete in speed with the all-analog processor. It must be emphasized that if higher basic frequency rates are used, or N is much larger than 8, the all-serial processor will require considerably larger time than the all-analog if state-of-the-art ADCs are used. The serial-parallel processor time will be considerably smaller than the all-analog time (for a basic $f = 1$ MHz), but it must be remembered that N ADCs would be necessary.

3.4 MOS Analog Multiplier Compatible with CCD Structures

The implementation of the analog TSVIP requires on-chip analog multipliers and op amps. The CCD structures use MOS technology, and consequently the design of the chip will be simplified if the multipliers and op amps use the same technology. Several NMOS high performance op amps have been reported in the literature, [9].

A four quadrant NMOS analog multiplier which seems adequate for our application was designed by Bosshart, [10] for a CCD signal processor. The basic structure of the multiplier is given in Fig. 3.7. The transistors are n channel MOS enhancement type devices. Recall that the output of CCD cells have dc bias level, and the inputs to the multipliers come directly from CCD taps. The voltages V_{IN1} and V_{IN2} will then vary with the signal about the dc bias level, but will never become negative. Q_1 and Q_2 are biased by V_{DD} , V_{SS}

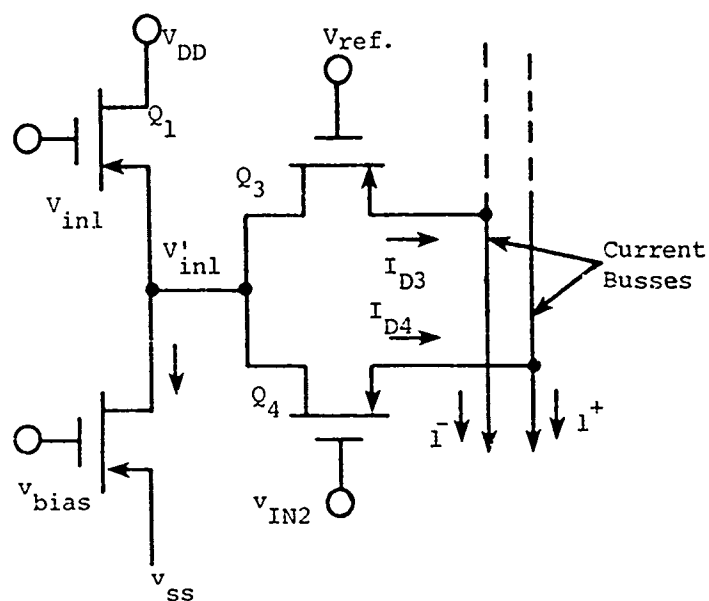


Figure 3-7 nMOS Multiplier Basic Cell

Note: $v_{in1} = v_{dc\ bias\ 1} + v_{in1}$

$v_{in2} = v_{dc\ bias\ 2} + v_{in2}$

and V_{bias} to work in the saturation region, Q_2 acting as a current source and Q_1 as a source follower. Q_3 and Q_4 must work in the "triode" region (the essentially linear region before saturation), and they work as voltage controlled resistors. Their V_T must be large. If the busses to which the sources of Q_3 and Q_4 are connected are at a potential equal to the dc bias at the output of the CCD devices, the V_{DS} for these two transistors will be equal to V_{in1} because the output voltage of Q_1 is practically the same as its input voltage.

For Q_3 and Q_4 the drain current is given by

$$I_{D3} = B (V_{ref} - V_T) V_{DS} - \frac{1}{2} V_{DS}^2 \quad (3.4.1)$$

$$I_{D4} = B (V_{IN2} - V_T) V_{DS} - \frac{1}{2} V_{DS}^2 \quad (3.4.2)$$

The summation of I_{D4} and $(-I_{D3})$ is then

$$I_{D4} - I_{D3} = BV_{in1} (V_{IN2} - V_{ref}) \quad (3.4.3)$$

If V_{ref} is made equal to the dc bias level of V_{IN2} , we see from (3.4.3) that

$$I_{D4} - I_{D3} = BV_{in1} V_{in2} \quad (3.4.4)$$

This basic multiplier produces distortion terms due to the high output impedance of the CCD devices which drive Q_4 , the output impedance of Q_1 , etc. It is possible to correct these errors by adding extra components to the circuit, such as a source follower between the output of the CCD and the gate of Q_4 , or the error terms could be corrected, as suggested in [10] by performing a weighted summation of I_{D4} and $-I_{D3}$. The only way of checking the performance of the device is by means of a prototype, although an extensive theoretical analysis taking all these factors into consideration could be performed. For a channel length of 1.5 μm , well within present technology limits, the maximum bandwidth of the device is 5 MHz.

3.5 Experimental Results for the Multichip Implementation of $[P_C]^+$

3.5.1 Introduction

Although the eventual goal of this research is to design and construct the CCD processor in a single chip or, if this is not possible, in a minimum number of chips, it was desirable at this time to test the feasibility of the CCD implementation by constructing one of the TSVIP blocks using commercially available CCD devices, multipliers and op amps.

With reference to Fig. 3.2, we see that the implementation of $[P_C]^+$ requires the use of two PISO CCD devices, two SIPO CCD devices, four analog multipliers, and two summers. The use of commercially available CCD devices plus the requirements presented by variable matrix operations, resulted in some problems which can be easily avoided when CCDs are used to implement transversal filters or discrete correlators.

To our knowledge only EG&G Reticon produces commercial SIPO and PISO CCD devices at present. These devices have an output dc bias level of about 9.5 V for the SIPO and 5 V for the PISO. The output signal appears superimposed to this bias. For the Reticon TAD-32 SIPO device, the input signal can have a maximum value of 3 V pp. If a higher amplitude is input, the positive swing of the outputs is clipped, due to the dc bias. For transversal filters and other single output applications, the dc bias can be eliminated at the output by means of a simple resistor-difference amplifier circuit, [11]. For our application, however, in which the N outputs have to be used individually, a dc bias elimination circuit would have to be used at each tap. In an integrated circuit processor this would not be a difficult requirement, because only a few transistors and resistors with values convenient for integration are needed. If discrete components are involved, as in our experimental circuit, the resultant circuit would be too bulky. A simple approach, not

convenient in practice, was used in our circuit, which will be explained in section 3.5.2.

Another problem is presented by noise introduced by the sampling process. Observations very similar to those made above apply also in this case. In transversal filters, the noise can be filtered once at the output. The sampling noise can be explained graphically by means of Figs. 3.8(a) and (b). In Fig. 3.8(a), the lower trace is a sinusoidal input at 500 Hz, sampled at 500 KHz. The upper trace is the output at tap 1. The noise is inherent to the sampled operation of the CCD in which a "piece" of the input is transferred at each clock pulse. The sweep in Fig. 3.8(a) is at 1 ms/div, while in Fig. 3.8(b) the same two signals are shown with the sweep at .5 μ s/div. The notch at the tap #1 output is due to the sampling-holding-transferring and shows up as noise.

3.5.2 Experimental Circuit

As mentioned in the previous section, the experimental circuit was constructed using multiple chips. Four vector boards were used, one for the PISO circuit, one for the multipliers/adders circuit, one for the SIPO circuit, and one for the clocks. The PISO devices are the Reticon R5501. As can be seen in Fig. 3.9, two bias voltages are required for these devices. The value of the bias voltages affects the maximum allowable peak to peak value of the signal. Bias voltages were adjusted for optimal operation. The R5501 requires three clocks, ϕ_1 and ϕ_2 which are the complement of each other and which work at the frequency of transfer, and ϕ_T which works at the loading frequency. One ϕ_T pulse is required for every 32 ϕ_2 pulses, because the device has 32 parallel inputs. This is in accordance with the F and NF frequencies indicated in Fig. 3.2. In a custom made device the number of parallel inputs would be N.

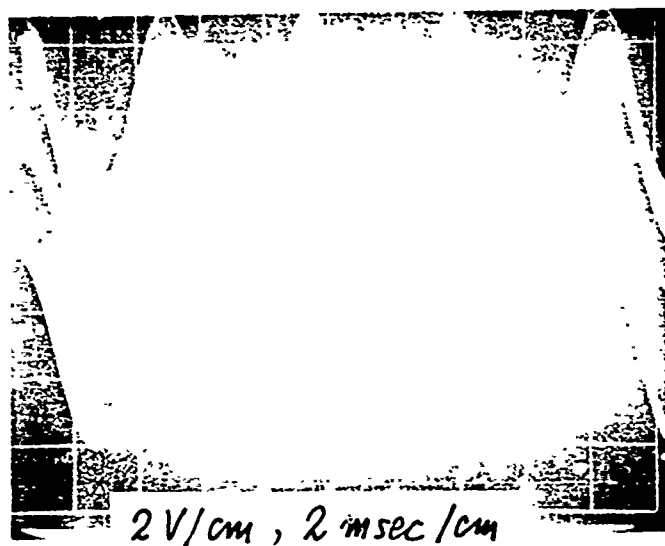


Fig. 3.8(a) Output At Tap 1 And Input, SIPO Device.

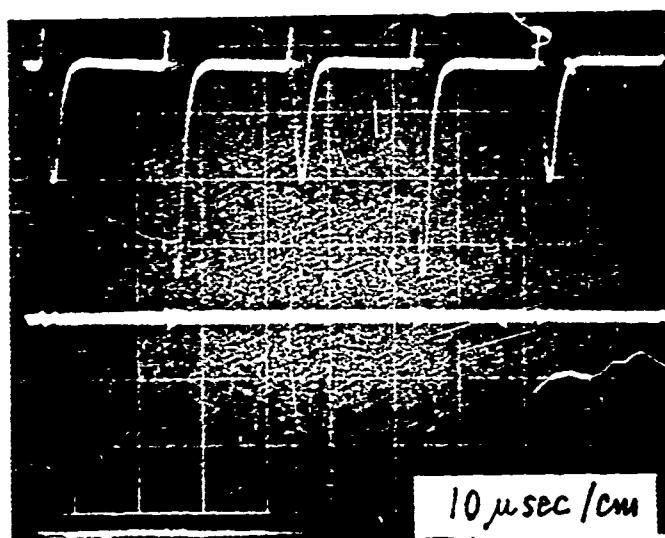


Fig. 3.8(b) Sinusoids Of Fig. 3.8(a) With A Much Shorter Sweep Time

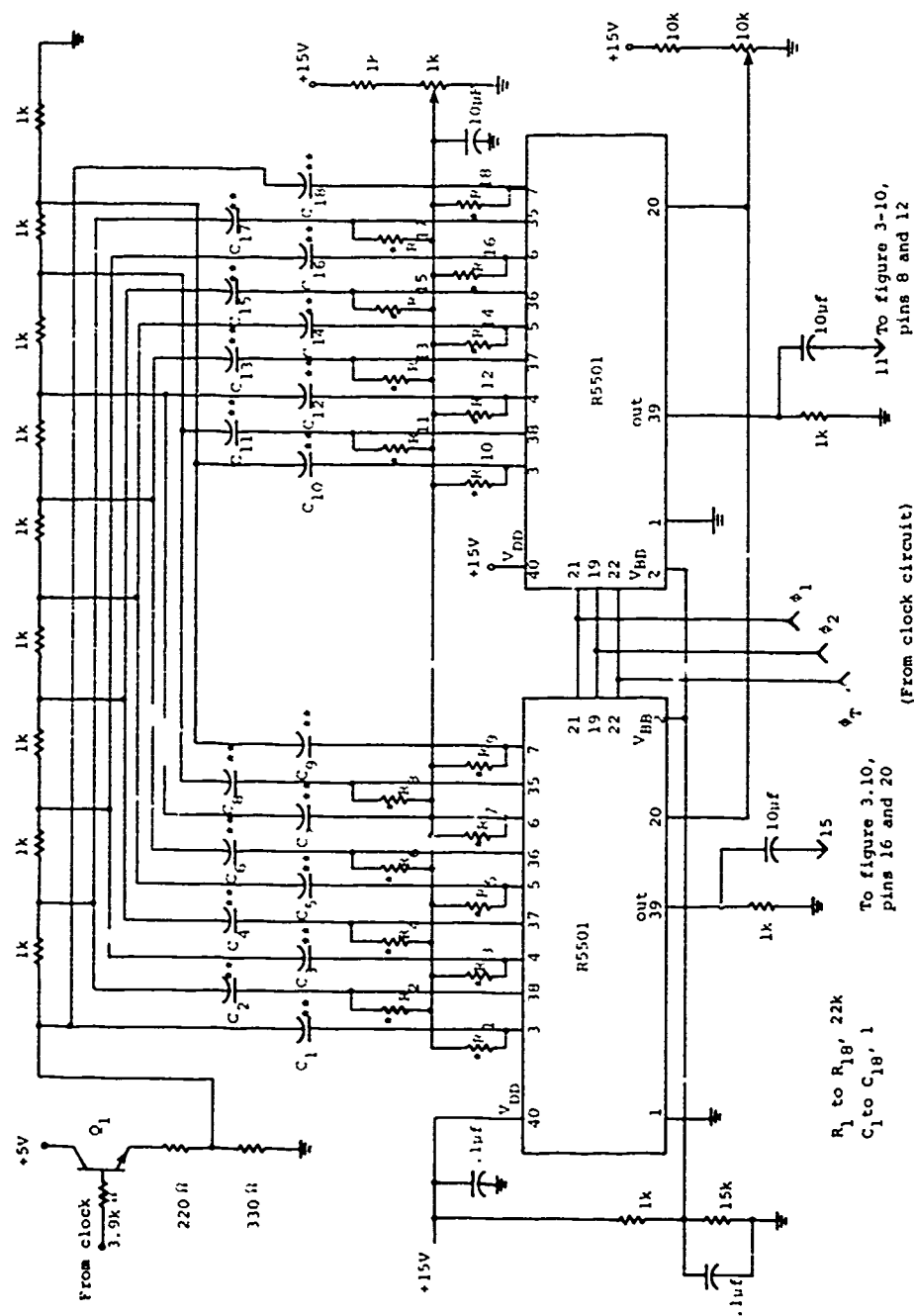


Figure 3.9 FISO Devices Circuit: Diagram

In the Reticon device loading is effected when ϕ_T and ϕ_2 are high, but ϕ_T must start at least 50 ns before and end at least 50 ns after the corresponding ϕ_2 pulse. A combination logic TTL circuit was designed to obtain ϕ_T , ϕ_1 and ϕ_2 from a single external clock signal. The signals are transferred entirely out of the CCD device during the next 32 clock pulses of ϕ_2 and ϕ_1 , when ϕ_1 is high and ϕ_2 is low. The sampled signals are first stored in the input capacitors, and then transferred into the delay line. This process has an effect in the output, as will be seen later.

The specified maximum frequency for the R5501 is 5 MHz, [12], although the performance of the devices degrades quickly for $f > 2\text{MHz}$. Consequently, NF was chosen to be 1 MHz, with $F = 31.25\text{ KHz}$. This is well below the frequency specified in the previous report, but it is the best achievable with available devices. In general SIPO devices perform better than PISO, and as was suggested in Sect. 3.2, it could be more convenient to use SIPO devices only in the SOP configuration to implement the processor. The lack of time and of the required parts prevented the testing of this architecture.

The SIPOs used in the experimental circuit also present the problem of sensitivity to bias voltages, and these were adjusted for optimal operation. The circuit for the SIPO CCDs is given in Fig. 3.11.

The multiplier chips chosen were the AD533, with a maximum error of less than .5% of full scale, and a small signal unity gain of 1 MHz. The summers were implemented using the AD 507J op amp which has a 100 MHz GBW product, a 25 V/us minimum slew rate, and a unity gain BW of more than 10 MHz. The circuit is given in Fig. 3.10.

With reference to Fig. 3.2, we see that the dc bias at the output of the PISO devices must be blocked before the signals are inputted to the multipliers.

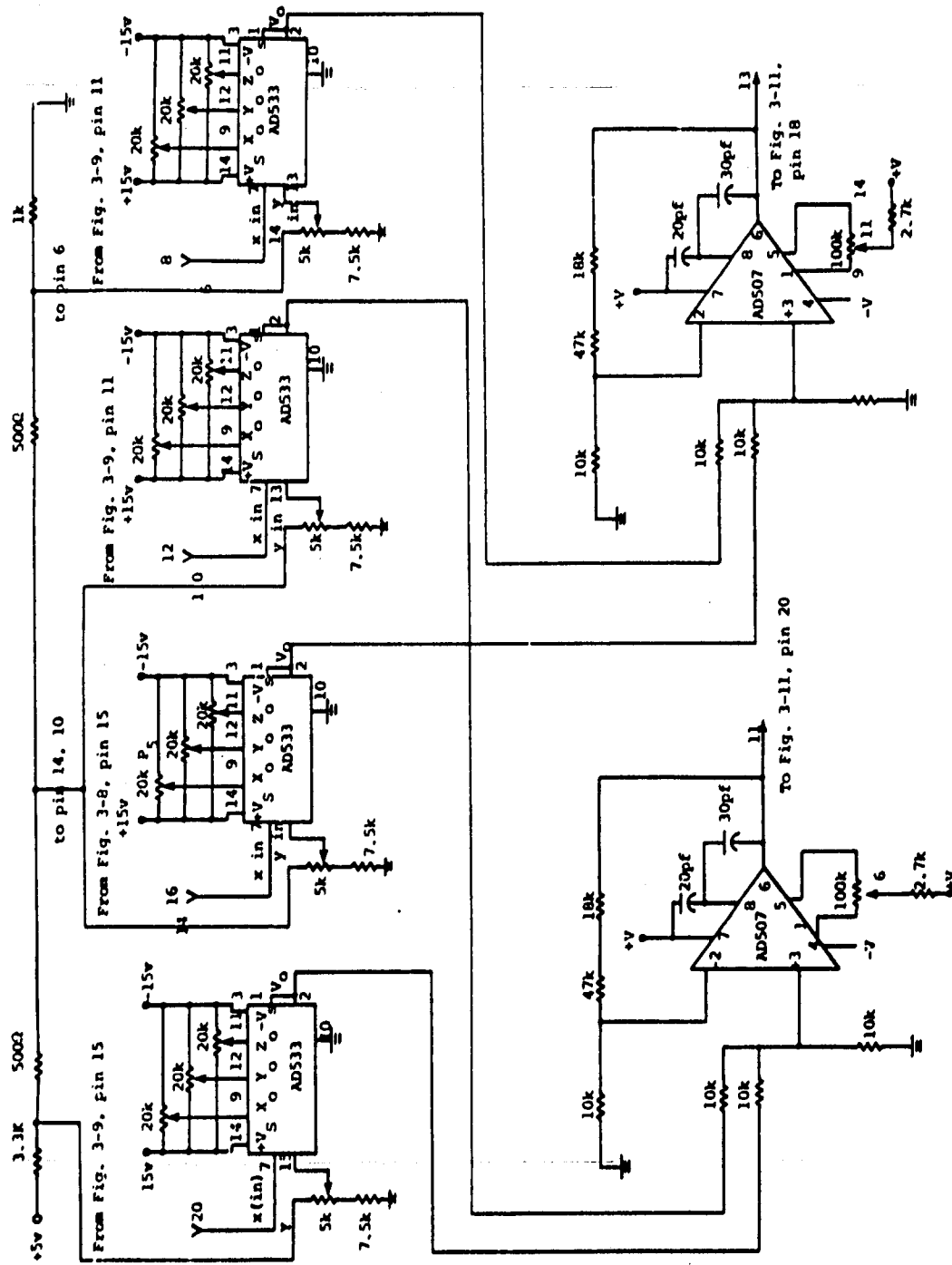


Figure 3.10 Multipliers and Summers Circuit Diagram

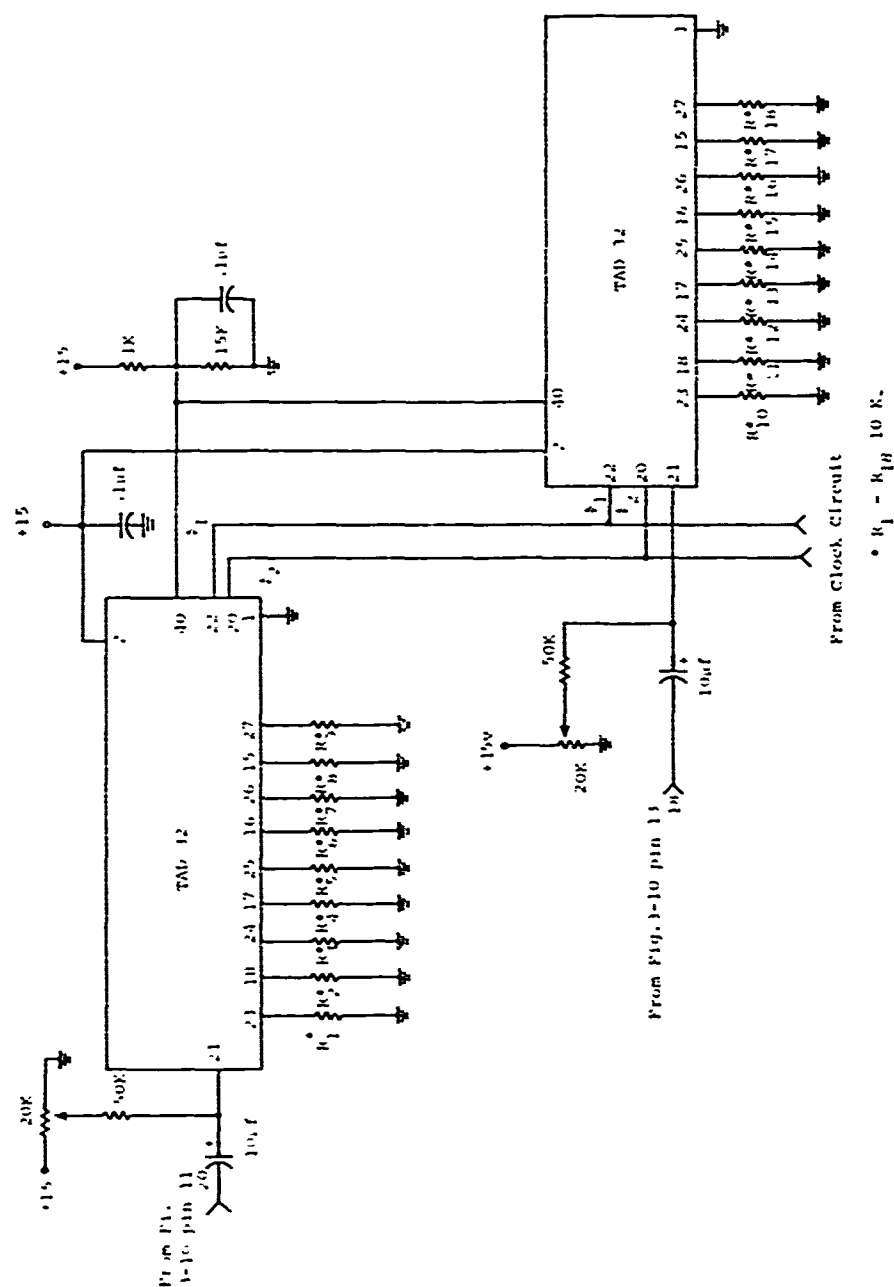


Figure 3.11 SIPO Device Circuit Diagram

As already mentioned, the discrete circuit implementation of a resistive network/differential amplifier circuit would be too cumbersome for this purpose. For this qualitative study, a simple blocking capacitor was used.

In order to test the devices qualitatively, a staircase voltage from -0.22 V to -2.0 V in steps of 0.22 V was applied to taps 1 through 9 of the PISO device No 1, and a staircase from -2 V to -0.22 V to the PISO device No 2 taps 1 to 9. With reference again to Fig. 3.2, q_{11} was chosen as 2 V dc, q_{12} as 1.5 V dc and q_{22} as 1 V dc. The input voltages to the first nine taps of the PISO devices can be expressed as follows,

$$V_1 = -mx \quad (3.5.2.1)$$

$$V_2 = -2 + mx \quad (3.5.2.2)$$

where $m = 2/9$ is the ramp slope and x is the tap number. Multiplication by the constant "q" inputs to the multipliers only changes the scaling of these voltages, and addition of the multipliers outputs produces another straight line staircase,

$$\begin{aligned} V_{M_1} + V_{M_2} &= -m_1 x - K + m_2 x \\ &= -K + (m_2 - m_1)x \end{aligned} \quad (3.5.2.3)$$

Waveforms for the experimental circuit are shown in Figs. 3.12 to 3.14. In Fig. 3.12(a), the upper trace shows the ϕ_T clock pulse. The lower trace shows a few ϕ_2 clock pulses. The time relation between ϕ_T and ϕ_2 shows clearly. Fig. 3.12(b) shows the output of PISO device No 1. This waveform corresponds to expression (3.5.2.1). The split of each of the nine steps into two levels is inherent to the operation of the R5501, and is due to the two step storage-transfer operation for ϕ_2 high and ϕ_1 high, respectively. The output of the multipliers should be filtered in the final version of the processor, in order

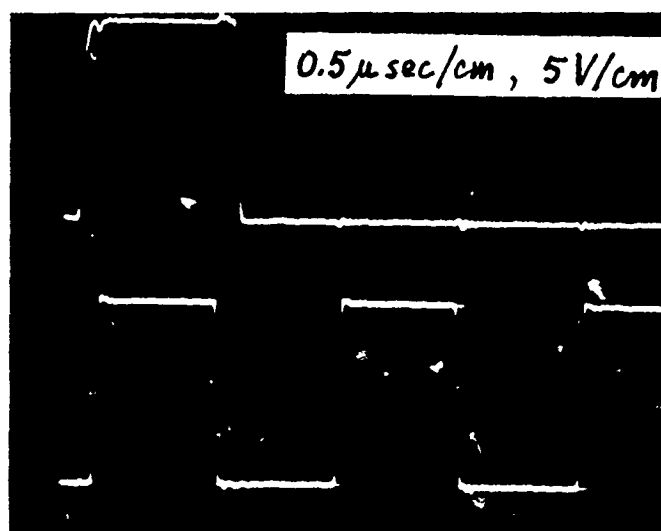


Fig. 3.12 (a) ϕ_T And ϕ_2 Waveforms 5V/cm; Sweep At 0.5 μ s/cm.

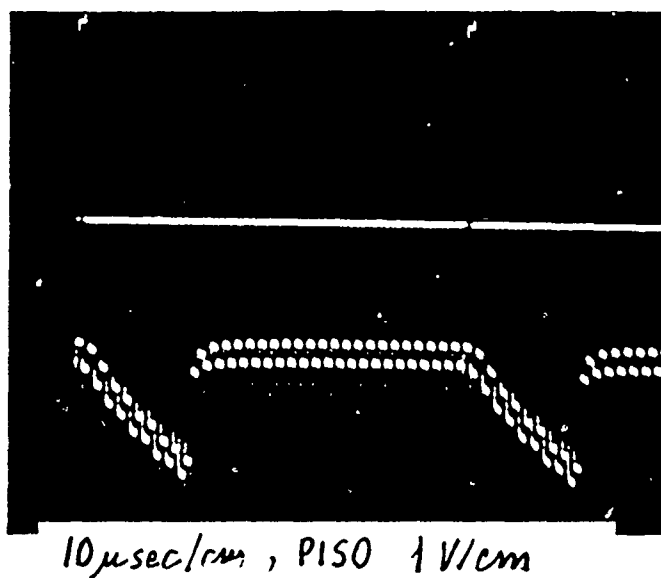


Fig. 3.12(b) ϕ_T Clock And Output Of PISO Device No 1.

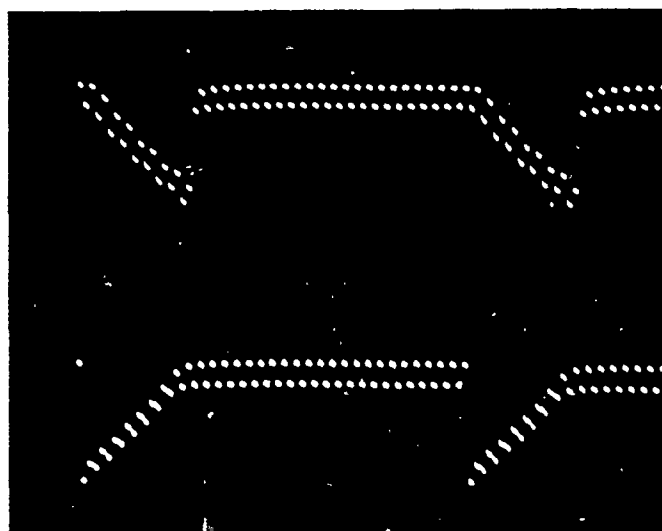


Fig. 3.13(a) PISO Devices Output

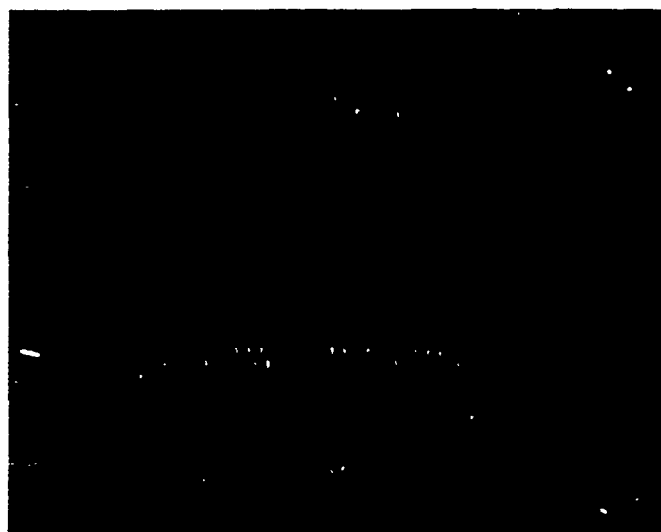


Fig. 3.13(b) First Two Multipliers Output

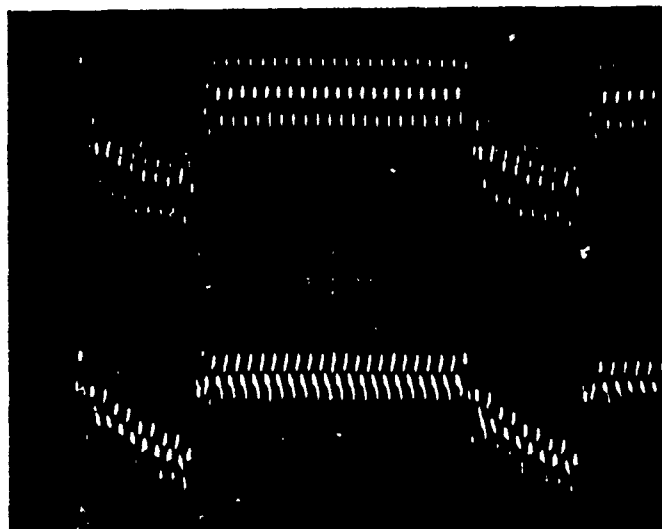


Fig. 3.14(a) Output Of Summers

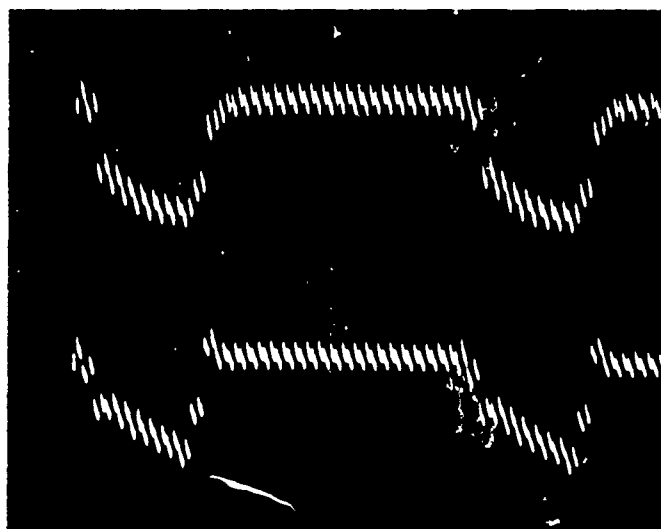


Fig. 3.14(b) SIPO Devices Output At Taps No 9.

to avoid higher noise and distortion in successive stages. Fig. 3.13(a) shows the outputs of PISO devices 1 and 2 (upper and lower traces, respectively). Fig. 3.13(b) shows the output of the first two multipliers of Fig 3.2. At this point noise becomes more evident due to the two different levels for each output pulse of the PISO devices. The waveforms have the correct shape, however.

Noise and distortion increase significantly at the output of the summers, Fig. 3.14(a). In addition to the sources of noise mentioned above, the use of several boards introduces grounding problems which would be eliminated in an improved version or in a single chip design of the device. Finally, in Fig. 3.14(b) which shows the output at tap 9 of the SIPO devices, although the general shape of the expected signal is preserved, noise and distortion have increased further.

In spite of the high noise and distortion levels, this experimental version of the pseudo inverse $[P_C]^+$ block succeeded in showing that it is possible to implement the TSVIP by means of CCD devices and other analog devices.

3.5.3 Conclusions and Recommendations

The main sources of error in the implemented block are:

- (a) Coupling capacitors which change the dc level at the input to the multipliers by an amount corresponding to the average value of the ac signal.
- (b) Noise due to sampling-storage-transfer in the CCD devices.
- (c) Ground problems due to the multi-board implementation, multiple power supplies and the necessity of three clocks.

This was a simple experimental version of the subsystem using readily available components and many discrete parts (resistors, capacitors, etc.). Noise and distortion could be drastically reduced in a system using specially

designed CCD devices and a high degree of integration. The use of SOP architecture which allows implementation using only SIPO devices (plus multipliers) should be considered, due to the higher performance of SIPO devices.

The Reticon devices are surface charge devices. The design should be based on buried channel ("peristaltic") devices, which do not need a "fat zero" (i.e., an input dc bias level) and have higher speeds, better charge transfer efficiency and a larger dynamic range than surface channel devices [13.14].

CHAPTER IV

THE USE OF SEQUENTIAL ESTIMATION TECHNIQUES IN IMPLEMENTING THE TS'IP ALGORITHM

4.1 Introduction

4.1.1 Background

The TSVIP algorithm reduces the problem of image tracking to a problem in linear estimation [5]. The method by which this linear estimation is performed determines the speed and ease with which the TSVIP algorithm can be implemented. A linear estimation procedure which can be performed rapidly, but is difficult or impossible to implement with the desired circuit technology is useless. Likewise, an estimation procedure which lends itself to easy implementation with the desired technology, but is temporally inefficient is also of no use.

Charge coupled devices (CCDs) are the devices with which it is desired to implement the tracking processor. As pointed out elsewhere in the report, CCDs have been selected because of their size, low power consumption, reliability and speed [14]. It is anticipated that an entire image tracker including a CCD imager could be constructed on one VLSI integrated circuit. While a clock rate of from 1 to 50 MHz would be sufficient for a tracking processor, using the solution approach discussed in this chapter, experimental CCDs have been clocked at up to 1 Gmz. Results of this sort certainly make it desirable to find a linear estimation procedure which lends itself to easy implementation with CCDs.

4.1.2 Closed Form Techniques

Two estimation procedures which meet the first criterion of speed are the matrix inverse and the generalized inverse applied to square and overdetermined systems, respectively [15]. The matrix inverse provides an estimate with no associated error, while the generalized inverse provides an estimate whose

associated total squared error is minimized. The matrix inverse is of little use since only overdetermined systems are anticipated. The main drawback of these techniques is that both require a matrix inversion which is difficult if not impossible to perform totally with CCDs (multipliers may be required). Therefore, it is necessary to look beyond these closed-form procedures for a suitable estimation procedure.

4.1.3 Sequential Procedures

Sequential procedures are a class of estimation procedures which satisfy the second of the aforementioned criteria. They call for just the sort of linear combinations of discrete-analog sequences that CCDs are able to provide. This makes sequential estimation procedures better suited to CCDs than the closed form procedures.

A sequential estimation algorithm is a recursive procedure which, using an initial "guess," converges to a solution of a system of a linear equations which minimizes a chosen criterion function. Most often the total squared error associated with the estimate is chosen for the criterion function. Sequential estimation procedures which minimize a squared-error criterion function can be shown to converge to the same solution as the generalized inverse as the number of recursions performed becomes infinite. For practical reasons, recursion is terminated after a reasonably accurate solution has been obtained. The usefulness of a sequential procedure is severely limited if a large number of recursions is required to arrive at an acceptable estimate of the solution to a system of linear equations. A judicious choice of the starting "guess" and the use of convergence acceleration techniques (to be subsequently discussed) can be of some help. However, the utility of sequential estimation procedures in implementing the TSVIP algorithm is extremely dependent on the speed with which the procedures converge.

4.2 Minimum Squared-Error Descent Procedures

4.2.1 Introduction

The sequential estimation techniques chosen for investigation are known as minimum squared-error descent procedures. These sequential estimation procedures use the gradient descent method to minimize the squared-error associated with the solution. Their suitability for CCD implementation makes them attractive for use in a target tracking processor.

The problem which must be solved is

$$d = Da \quad (4.2.1.1)$$

where

d is the $N \times 1$ scene difference vector,

D is the $N \times 6$ matrix of weighted and unweighted spatial derivatives,

a is the 6×1 vector of the affine transform coefficients (the quantity it is desired to estimate), and N is the number of target points used in the tracking calculation.

An estimate of the a vector which minimizes some function of the error between Da and d is sought. An error vector can be defined as

$$e = Da - d \quad (4.2.1.2)$$

The square of the total length of the error vector is then given by

$$\begin{aligned} J(a) &= \|Da - d\|^2 \\ &= \sum_{k=1}^N (a^t D^k - d^k)^2 \end{aligned} \quad (4.2.1.3)$$

where

D^k is the transpose of the k^{th} row of the D matrix,

d^k is the k^{th} element of the d vector,

a^t is the transpose of the affine transform coefficient vector.

It is this function which we wish to minimize in obtaining a "good" estimate of the vector a . A gradient descent procedure can be used to produce a sequence of vectors which will eventually converge to a solution minimizing $J(a)$. The form of a general gradient descent (also known as steepest descent) is given by

$$a_{k+1} = a_k - p_k \nabla J(a_k) \quad (4.2.1.4)$$

where

a_k and a_{k+1} represent the k^{th} and $k+1^{\text{th}}$ estimate of a , respectively.

p_k is a positive scale factor which adjusts the step size [16].

The initial estimate a_0 is chosen arbitrarily unless a priori information is available to provide a reasonable guess. At each iteration of the descent procedure a fraction of the error gradient is subtracted from the previous estimate of a . If a_k is thought of as a point in the multi-dimensional a vector space, then the subtraction of a fraction of the error gradient represents a movement, in the vector space, in the direction of the maximum decrease in the magnitude of the chosen error criterion. Thus, each step is thought of as a downward movement in the direction of the location of the a vector which minimizes $J(a)$. Thus, the name "descent procedure."

4.2.2 Multi-Sample Algorithm

In this case the gradient of the criterion function $J(a)$ is given by

$$J(a) = \sum_{k=1}^N 2(a^t D^k - d^k) D^k = 2D^t (Da - d) \quad (4.2.2.1)$$

Inserting this result into equation (4.2.1.4) gives the descent algorithm

$$a_{k+1} = a_k - p_k D^t (Da_k - d) \quad (4.2.2.2)$$

It can be shown that if p_k is chosen to be p_1/k , where p_1 is a constant usually less than 1, this descent algorithm, which will be called the multi-sample algorithm, satisfies

$$D^t (Da - d) = 0 \quad (4.2.2.3)$$

Thus, this descent algorithm will determine a coefficient vector a which minimizes the total-squared length of the error vector.

An algorithm of this sort does fulfill the requirement of not requiring a matrix inversion. The multi-sample algorithm also satisfies the speed of convergence requirement previously discussed. Its main drawback is that its CCD implementation, while not impossible, would be very hardware intensive. The size of the D matrix, N , would determine the number of tapped delay lines (the basic CCD "building block") required to implement this algorithm. Thus, the number of target points used in tracking would determine the size of the processor. This is not desirable and necessitates looking further for a more suitable version of the descent procedure.

4.2.3 The Widrow-Hoff Rule

The error criterion used in arriving at the multi-sample descent algorithm represents, except for a constant multiplier, the average squared length of the error vector associated with estimate a_k . The movement at each iteration is in the direction of the maximum decrease in the average length of the squared error vector. Widrow and Hoff [17] introduced the idea of substituting individual error vectors for the average error vector. This allows the rows of the D matrix to be considered individually and leads to the LMS or Widrow-Hoff Rule

$$a_{k+1} = a_k + p_k (d^k - a_k^t D^k) D^k \quad (4.2.3.1)$$

This descent procedure allows the rows of the D matrix to be considered sequentially. If p_k is taken as p_1/k , p_1 a constant, the Widrow-Hoff Rule will tend towards a solution which minimizes the squared-error in the estimate of a_k . Moreover, the CCD implementation of the Widrow-Hoff Rule need consist of only one tapped delay line with associated multipliers, summers and steering

logic [18,19]. A block diagram representation of a possible CCD implementation is shown in Fig. 4.1. The multipliers, summers, and logic necessary to complete the circuit could be implemented on the same substrate as the tapped delay line. The tradeoff one must make for these advantages over the multi-sample algorithm is a reduced speed of convergence. The averaging of the error gradients used in the multi-sample descent algorithm serves to "smooth" the error gradients and reduce the effect of bad data, i.e., those individual error gradients which, because of noise or other random effects, do not represent a move towards the minimum of the criterion function. The Widrow-Hoff Rule discards this averaging and thus follows a less direct path to the minimum of the criterion function. This leads to reduced speed of convergence.

4.2.4 Convergence Acceleration

Iteration speedup techniques can be used to increase the rate of convergence of the Widrow-Hoff Rule [20]. A very effective method of accelerating its convergence is to hold p_k constant as long as the error term,

$$(d^k - a_k^t d^k) \quad (4.2.4.1)$$

maintains a constant sign. The reasoning behind this is as follows. When the search for the minimum of the error criterion is far from that minimum large steps at each iteration are desirable. As long as the sign of the error term is constant it is assumed that the search is far from the minimum, p_k is held constant and the steps stay the same size. When the error term changes sign this is an indication that the search has reached the vicinity of the sought for minimum. A smaller step size is then desirable to allow the search to converge more easily to the location of the a vector whose associated squared error is minimized. This is analogous to slowing down one's automobile so as to not pass by one's destination as it is neared.

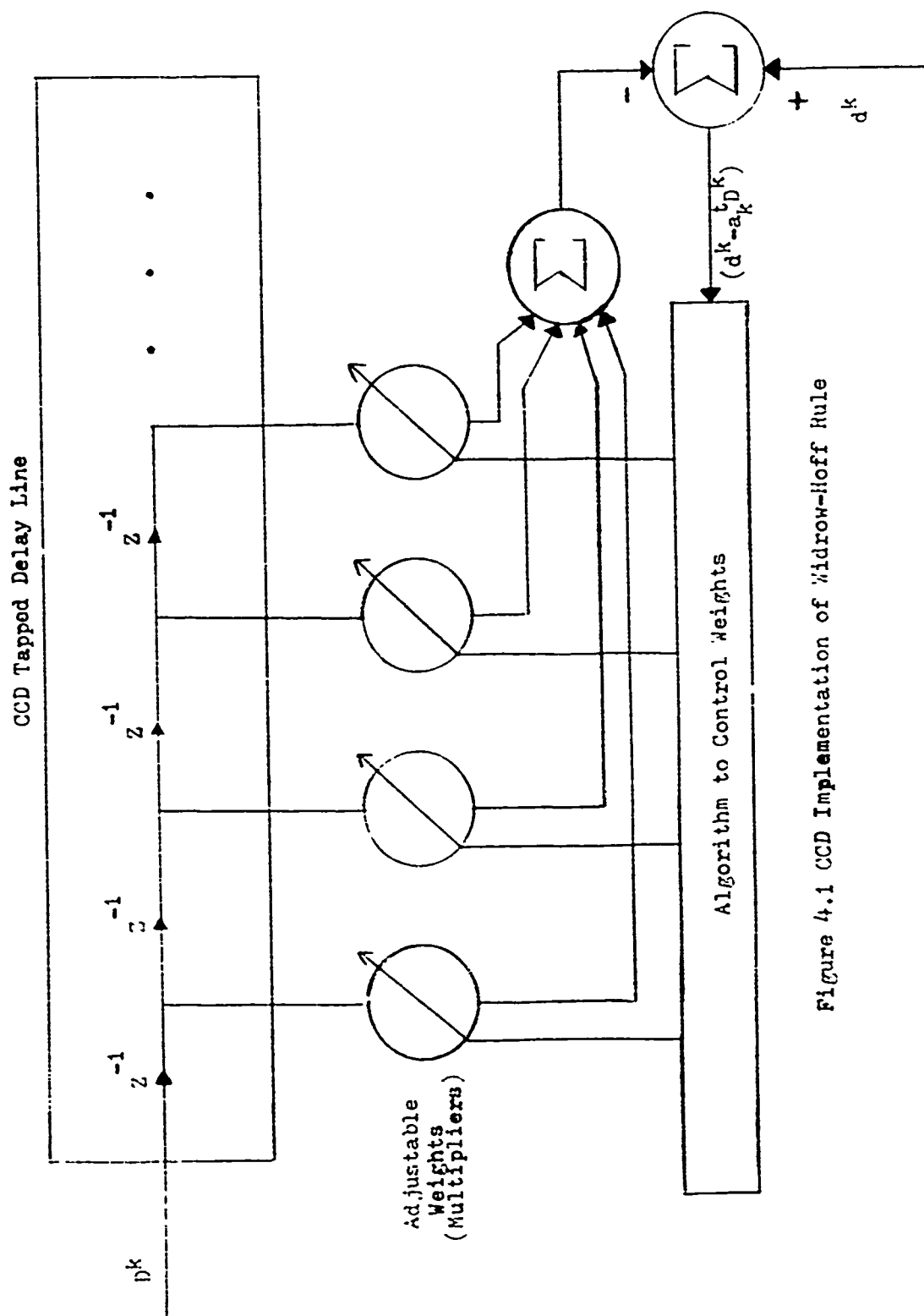


Figure 4.1 CCD Implementation of Widrow-Hoff Rule

4.2.5 Startup

The scene-to-scene temporal dependence required by the TSVIP algorithm provides another reason to be optimistic about the applicability of sequential descent algorithms to its solution. Both sequential methods presented, the multi-sample and Widrow-Hoff rules, require a starting "guess" of a_k . Without a priori information this is often chosen as 0. A more logical starting point, in this case, is simply the estimate arrived at for the previous camera frame. The assumption of fast temporal sampling required for the relevance of the TSVIP algorithm means that from frame to frame the target will appear to have a slowly changing velocity. This implies that the best "guess" for the present target movement is simply the previous estimated target movement. Using the previous target movement as the starting point for the next set of iterations provides a starting point, in the a vector space, which can be logically assumed to be near to the location which minimizes the squared-error associated with the estimate a_k . Since this allows the search to start closer to the minimum, the rate of convergence will be significantly accelerated.

4.3 Simulation

4.3.1 Introduction

A computer simulation was performed to ascertain whether the speed of convergence of the Widrow-Hoff Rule was sufficient to allow its use in implementing the TSVIP algorithm. Simulation, while not as realistic as the use of actual scene data, makes for easy confirmation of the properties of the algorithm. By varying one parameter, such as the convergence constant p_0 , while holding the rest constant, the effect of each parameter on the performance of the algorithm can be determined.

The simulated scene formulation is as found in [5]. The target textural function is given as

$$f(i,j) = \frac{\sin i\Delta + 1}{i\Delta} \frac{\sin(j\Delta/2.5)}{(j\Delta/2.5)} + .3j\Delta + 2 \quad (4.3.1.1)$$

where Δ , the sampling interval, is taken as .2. The target is taken as a square $16\Delta \times 16\Delta$, i.e., 16 pixels square. A flow chart of the Widrow-Hoff Rule implementation is shown in Fig. 4.2. The details of the simulated scene setup and those operations required to arrive at the linear system necessary for use of the Widrow-Hoff Rule can be found in [5].

4.3.2 Convergence Acceleration

The convergence acceleration technique used is a slight modification of the previously mentioned technique. The conventional technique is to use $p_k = p_0/k$ where p_0 is a constant and k is an integer index which is one initially and incremented by one each time the error term of the Widrow-Hoff Rule changes sign. The rationale behind this rule was previously stated. In simulation it was found that the convergence acceleration provided by this technique was not sufficient. Therefore the technique was modified by incrementing k only after a fixed number of error term sign changes, usually from three to five. It was found that very little if any further acceleration was provided beyond five sign changes. In this manner p_k did not decrease as rapidly and the step sizes, in the search for the optimum a vector, did not decrease as rapidly when compared to the step sizes produced with the conventional acceleration procedure. Through the use of this modified acceleration technique, the convergence of the Widrow-Hoff Rule could be accelerated by from 20 to 35%.

4.3.3 Size of Perturbation

The first property investigated was the effect of increasingly larger target perturbations on the accuracy and rate of convergence of the Widrow-Hoff procedure. Target movements of .05, .08, .1, and .2 were used. These represent movements of one-fourth, two-fifths, one-half, and one pixel, respectively.

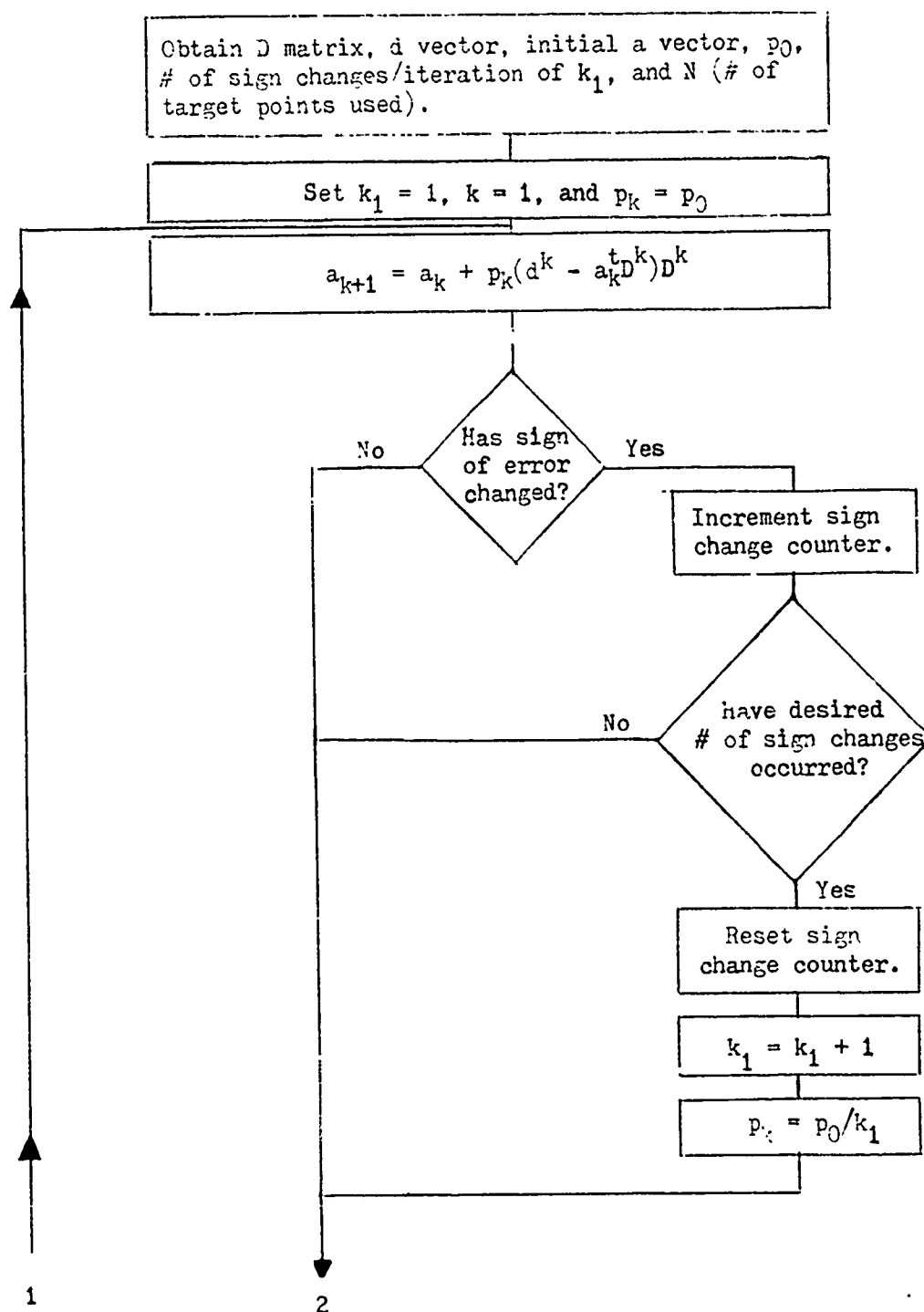


Figure 4.2 Flowchart of Implementation of Widrow-Hoff Rule

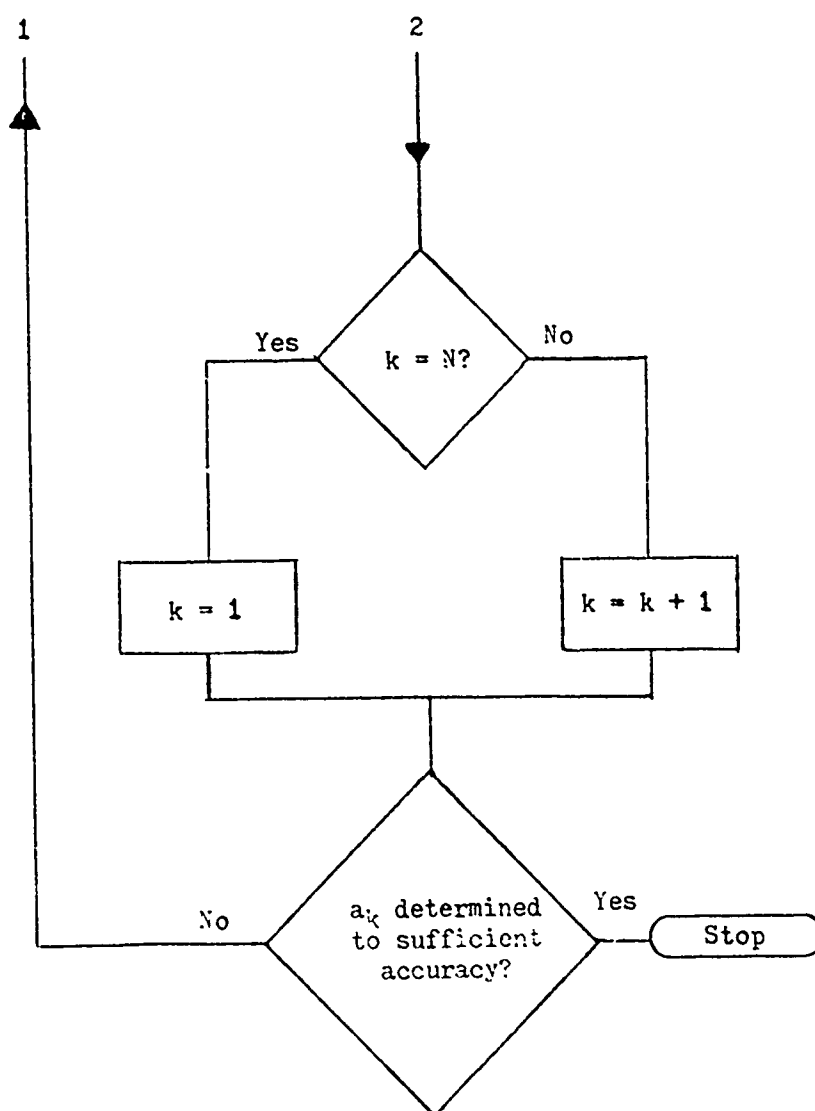


Figure 4.2 (continued)

The results of these simulated movements are summarized in Figs. 4.3-4.6.

The curves with a subscript of one indicate simulations in which the starting point was selected as zero. Those curves subscripted two indicate simulations in which a starting point which was from ten to twenty percent different from the actual target perturbation was selected.

One can see at once that the Taylor series approximation begins to become invalid as the size of the target perturbation increases. While the estimate of the $B(1)$ element of the translational vector remains accurate through the range of perturbations, the estimate of the $B(2)$ element falls from 95% accuracy for a perturbation of .05 to 80% accuracy for a perturbation of .2. For larger target perturbations than those shown, the $B(1)$ element also falls off in accuracy. This condition of more degradation of the $B(2)$ element of the translational vector than of the $B(1)$ element, for increasing target perturbation, is a function of the target textural function which is non-symmetrical in the $B(1)$ and $B(2)$ directions. The basic spatial frequency of the textural function in the $B(2)$ direction is two and one-half times that in the $B(1)$ direction. Thus, any attempt to express it in a truncated series must suffer from more truncation error than a similar expansion of the textural function in the $B(1)$ direction.

Figures 4.3 - 4.6 also provide valuable information about the selection of the initial convergence constant p_0 . In obtaining these curves, an attempt was made to use the largest p_0 possible while avoiding overshoot. Choosing p_0 too small resulted in extremely slow convergence of the algorithm. Conversely, choosing p_0 too large resulted in an oscillatory overshoot which also caused a slowing of convergence. The important finding is that the optimum p_0 increased as the target perturbation was increased. Thus, for optimum convergence p_0 cannot be chosen as a constant but must be adjusted from frame to frame.

Trans. Vector vs. # of Iterations (no dilation or rotation)

Perturbation $B(1) = B(2) = .25$
 1. Starting pt. = 0,0; $v_0 = .1$
 2. Starting pt. = .04, .04; $v_0 = .1$ } # of sign changes/iteration = 5

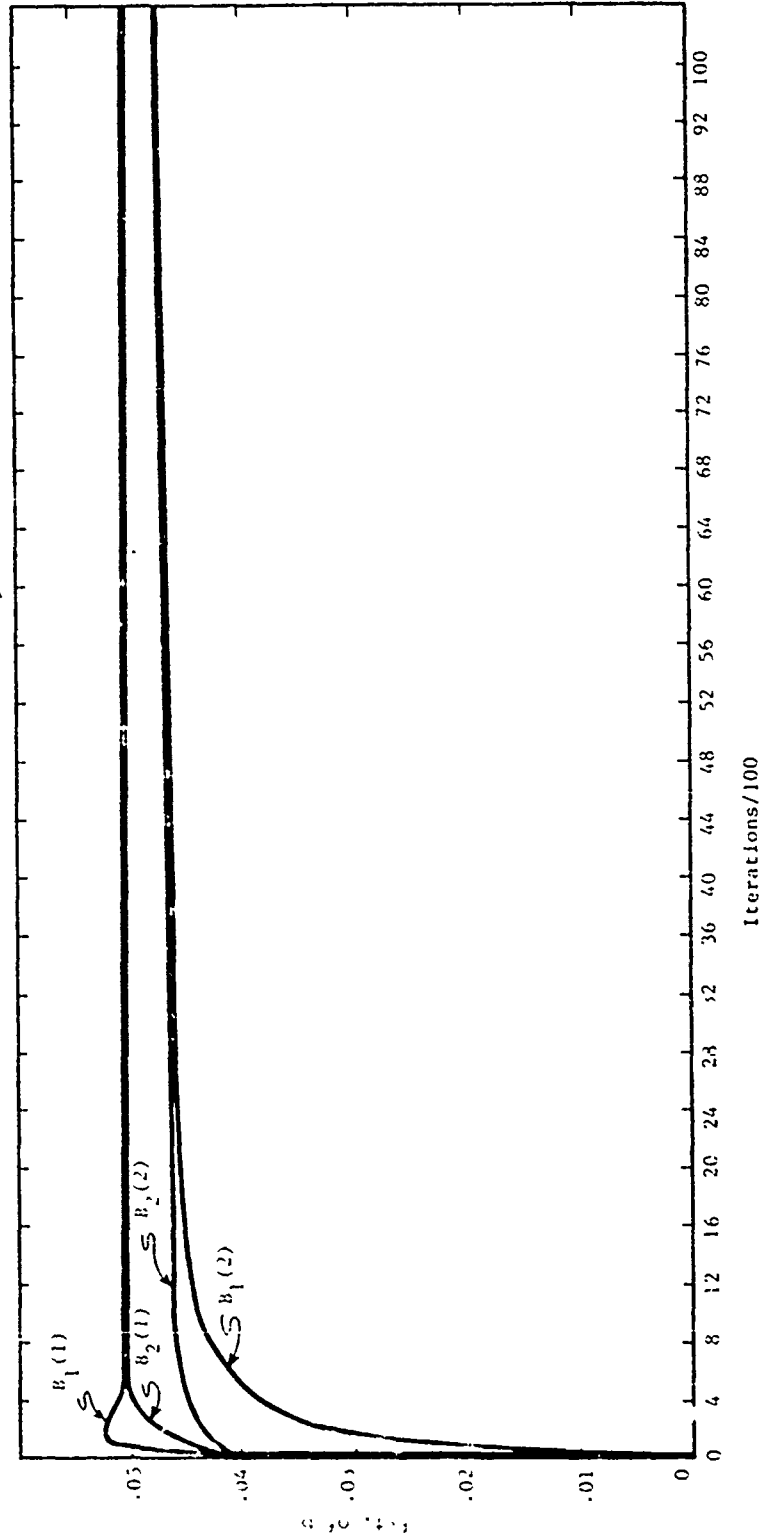


Figure 4.3

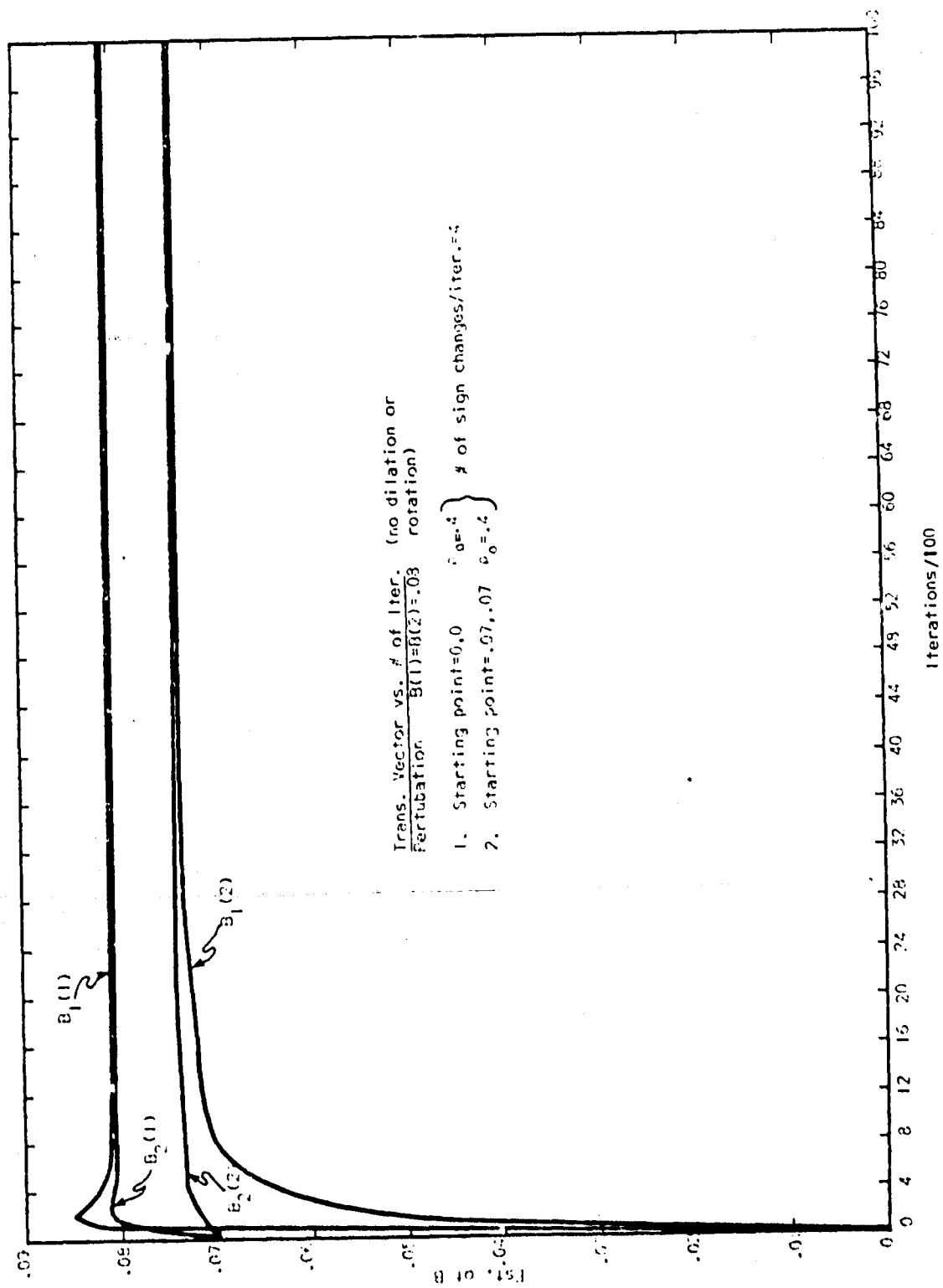


Figure 4.4

Trans. Vector vs. # of iterations (no dilation or rotation)
 Perturbation $B(1)=B(2)=.1$
 1. Starting pt. = 0,0; $\rho_0 = .4$
 2. Starting pt. = .08,.08; $\rho_0 = .55$ } # of sign changes/iteration=4

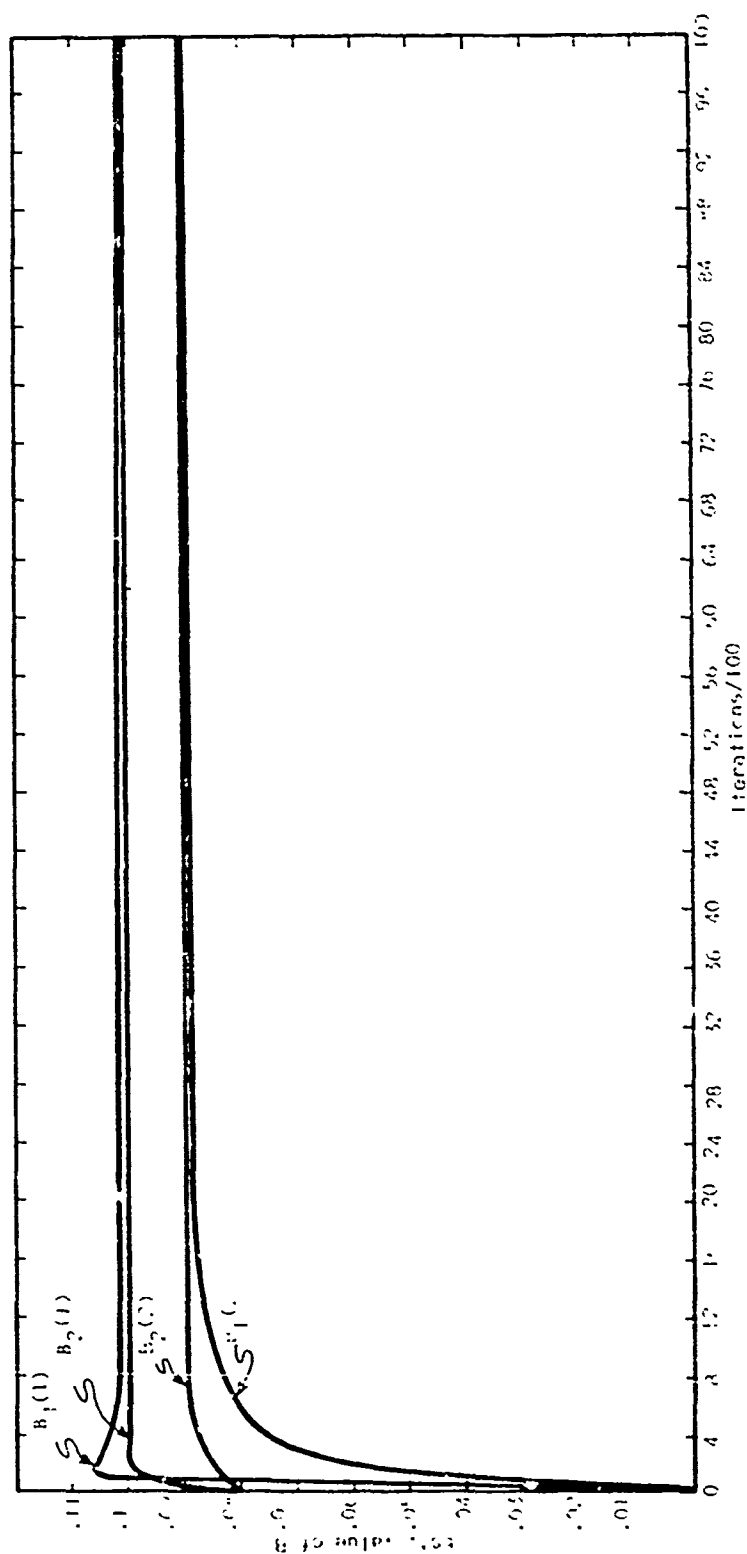


Figure 4.5

Trans. Vector vs. # of Iter (no dflation or rotation)

Perturbation = $B(1) = B(2) = .2$

1. Starting pt. = 0,0; $\nu_0 = .5$
 2. Starting pt. = .18, .18; $\nu_0 = .5$
- # of sign changes/iter = 4

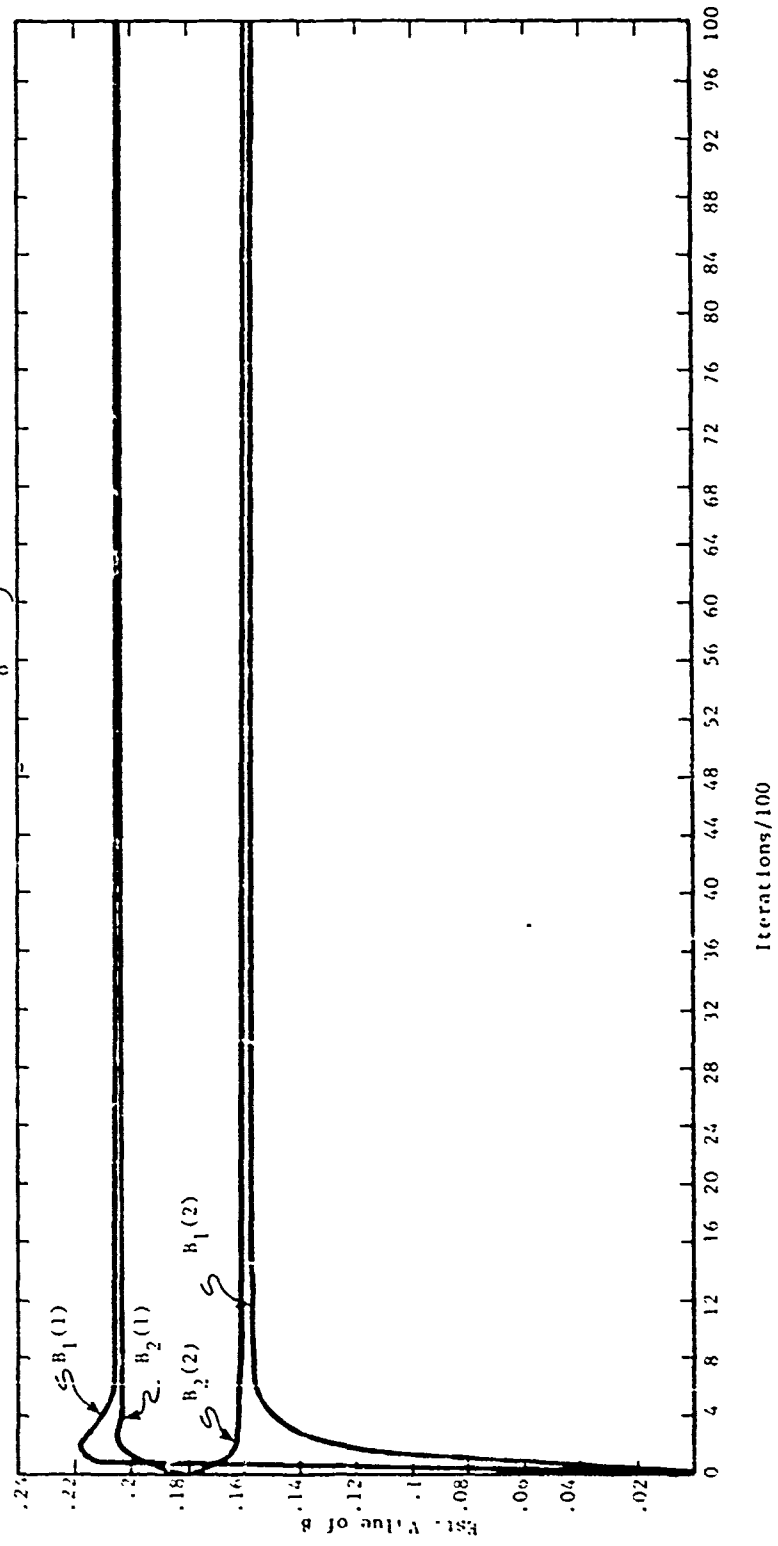


Figure 4.6

Since it is impossible to know a priori the magnitude of the target movement which is being estimated, the previous target movement is the best value on which to base a choice of p_0 . The assumption of fast temporal sampling inherent in the TSVIP algorithm should make this a reasonable assumption. In any case the choice of p_0 should be conservative to avoid any chance of overshoot or instability in the estimation process.

For completeness, a simulation which includes target dilation and rotation is summarized in Fig. 4.7. For small perturbations the estimates of both dilation and rotation were found to be accurate and to converge quite rapidly. A more exhaustive investigation was not carried out because dilation and rotation are not of primary interest in this investigation. That is, in a tracking processor, they are of secondary interest to the target translational parameters. Additional work will be required if they are needed for control purposes.

4.3.4 Startup

As was stated in the derivation of gradient descent procedures, a starting "guess" is necessary to compute an initial error gradient. The claim was made that the previous target translation was a good candidate for this starting point as the scene-to-scene temporal dependence inherent in the TSVIP algorithm guaranteed small changes in target velocity from frame to frame. To verify this claim the four increasingly larger target perturbations of Figs. 4.3 - 4.6 were run with starting points of from ten to twenty percent difference from the actual perturbation. The reduction in the number of iterations required for convergence ranged from 20 to 50%. The larger target perturbations benefitted more than the smaller from this choice of a startup point. The results indicate that an intelligent choice of the startup point for the Widrow-Hoff Rule can produce significant savings in the number of iterations required for convergence.

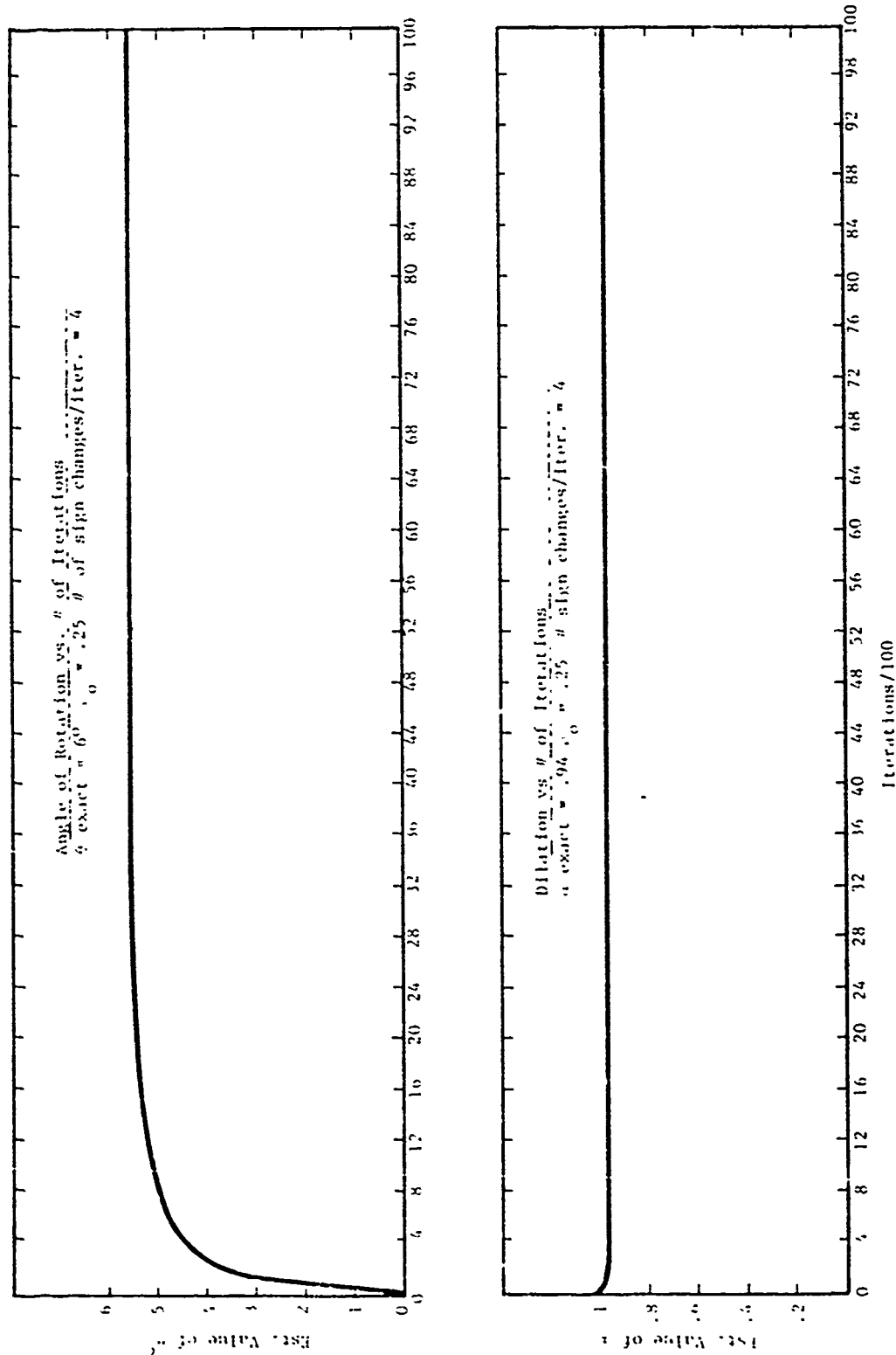


Figure 4.7

4.3.5 Noise Susceptibility

The previously discussed simulations were performed using exact spatial derivatives obtained from the target textural function and used no additive noise. To investigate the behavior of the Widrow-Hoff Rule in implementing the TSVIP algorithm using noise corrupted scene data the positive half of a normally distributed, zero mean noise process was added to the simulated scene. The spatial derivatives necessary to form the system of linear equations provided by the TSVIP algorithm were performed using a simple difference equation. The results are displayed in Fig. 4.8. The addition of noise to the scene produced rather unreliable estimates of the target perturbation. The estimation of spatial derivatives in a noisy environment is known to be unreliable. The error associated with additive scene noise can be attributed to this unreliability. The use of a 3 x 3 window to introduce some averaging into the spatial derivatives failed to produce any significant improvement in the results.

4.3.6 Summation

The main conclusions resulting from the preceding tracking simulations are as follows:

1. Larger target perturbations produce worse translational estimates than smaller perturbations due to the increasing truncation error inherent in a Taylor series approximation.
2. To achieve the same rate of convergence for larger target perturbations as for smaller requires an increase in p_0 , the initial convergence constant.
3. An intelligent choice of the starting "guess" in the Widrow-Hoff Rule, utilizing a priori information, can significantly improve the rate of convergence.
4. The estimation of spatial derivatives in a noisy environment introduces significant error into the estimated target translational parameters provided

Trans. Vector vs. # of Iterations w/added Noise
 Perturbation $B(1)=.05$ $B(2)$, not plotted, yielded similar results
 starting point=.04

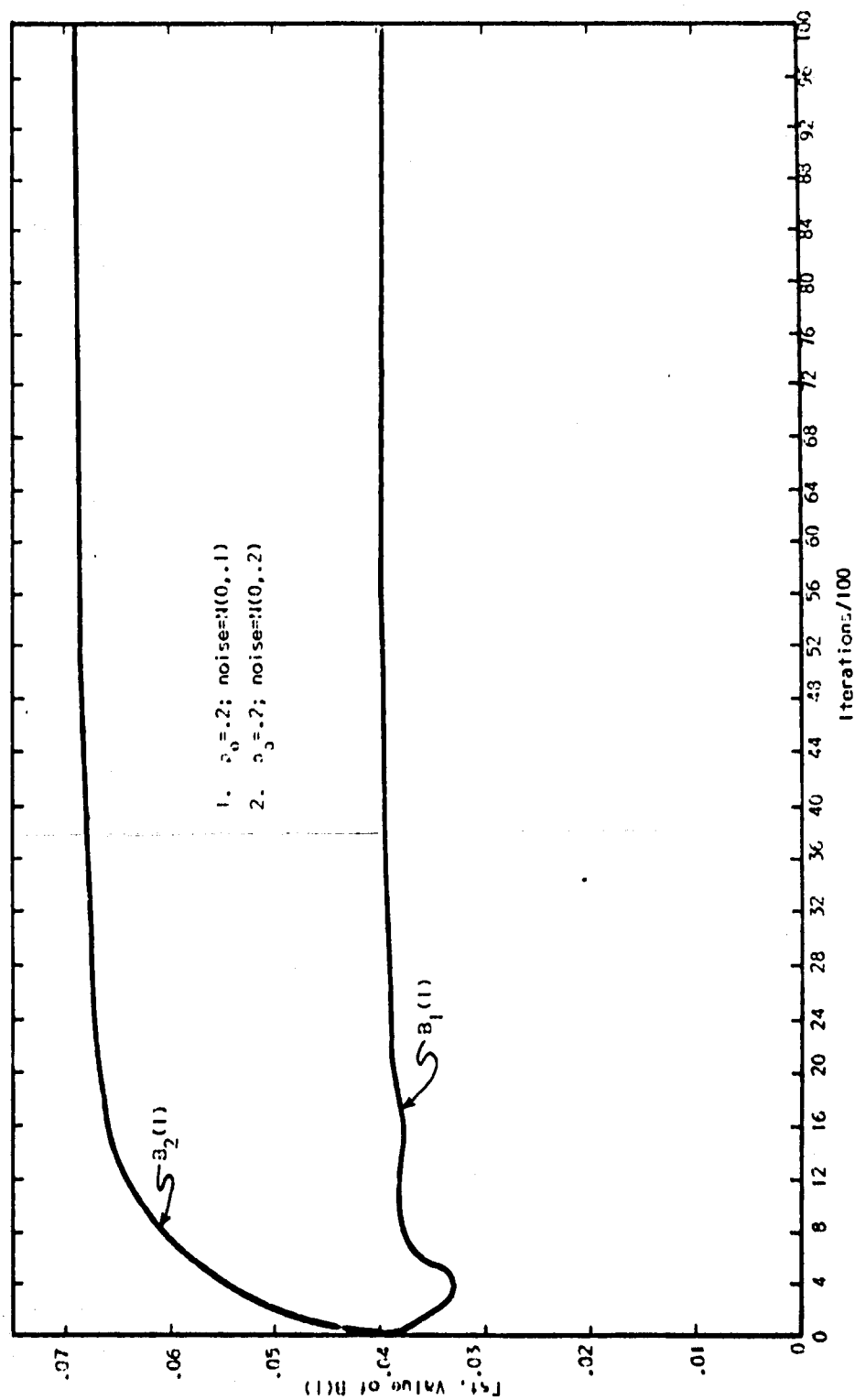


Figure 4.8

by the Widrow-Hoff Rule.

5. The results of these simulations are encouraging enough to recommend that the Widrow-Hoff Rule be tried in an implementation of the TSVIP algorithm using actual video data.

4.4 Conclusions and Recommendations for Future Study

The use of a particular sequential estimation procedure, the Widrow-Hoff Rule, has been applied to the implementation of the TSVIP tracking algorithm. For small perturbations and no additive scene noise, this estimation procedure has been shown to be effective in simulation. The use of a priori information and a slightly modified convergence acceleration procedure result in the convergence of the Widrow-Hoff Rule in an acceptably small number of iterations. The addition of noise to the simulated scene results in a significant degradation of the performance of the estimation procedure. This is a cause for some pessimism over the usefulness of the Widrow-Hoff Rule in the implementation of the TSVIP algorithm.

Future study will mainly involve obtaining actual scene data from an imaging device. It is hoped that the use of strict segmentation measures in defining target points will yield a system of linear equations which is well behaved, i.e., will yield accurate estimates when operated on by the TSVIP algorithm.

CHAPTER V

OTHER ALGORITHMS

5.1 Introduction

5.1.1 Purpose

This section reports on an investigation into the other algorithms for image tracking. Although the TSVIP algorithm appears to be the most promising, an investigation into other algorithms is necessary for several reasons. First, should the TSVIP approach prove to be ineffective, a list of possible alternatives would have been studied. Second, although the TSVIP method may work well as long as the target remains locked within the field of view, the overall system performance may require a hierarchy of algorithms which will be used in support of the TSVIP algorithm. For example, algorithms which can segment the target from the background, and ones which can locate and be used to lock the target within the field of view, may be required in some applications. Finally, alternative algorithms can serve as a basis of comparison to show how well the TSVIP algorithm performs. Thus, the general nature of this investigation is to discover image processing techniques which can either support or replace the TSVIP algorithm.

5.1.2 Constraints

There are several constraints on the algorithms that can be considered. These constraints are related to special performance and hardware requirements for the expected range of applications. The algorithms will be expected to perform in a real-time closed-loop system, and they must provide adequate lock-in and tracking performance at extremely high data rates. Thus, the algorithms themselves need to be very efficient; that is, both fast and accurate.

There are also several constraints on the type of hardware that can be implemented. The hardware will be subjected to large forces, will need to fit in small places, will probably be powered by a small power source, and will process data at a high rate. Therefore, the technology used must provide hardware that will be rugged, small, lightweight, low powered, and fast. The appropriate choice appears to be CCD based discrete analog processing, a conclusion reached in the previous algorithm work.

In CCD based image processing the analog signal is sampled at discrete intervals in time and space and processed as a discrete analog signal. As shown in a previous section, CCD devices can perform complex signal processing operations at high data rates while retaining a small size with low power consumption. But more importantly, the operations it performs are adaptive. That is, the filter transfer function can be altered in response to the data it receives. This allows a flexible and powerful hardware structure where the same basic elements can be used to perform different functions.

From the above discussion it is apparent that the algorithm search needs to be limited to those algorithms that can take advantage of CCD based structures. More explicitly, the bulk of the high speed processing should be performed with discrete analog technology implemented with CCDs, while the slower and simpler processing may use both digital and analog technologies. This hybrid of technologies will help to ensure high data processing rates, ruggedness, low power, small size, and low weight.

Constraining the hardware to be based largely on discrete analog processing unfortunately creates a constraint on the kinds of algorithms that can be investigated. There is basically one operation that is ideal for discrete analog processing, and any algorithm that can be considered in

this investigation should use this operation extensively. This operation is given mathematically as

$$y(k) = \sum_{i=0}^{N-1} a(k,i)x(k-i) \quad (5.1.2.1)$$

where $y(k)$ is the discrete analog output at time k , $x(k)$ is the discrete analog input at time k , and $a(k,i)$ is the i^{th} coefficient for the operation at time k . Notice that the output is simply a weighted sum of delayed input values. It is often referred to as a transversal filter, a moving average filter, or a finite impulse response filter.

By the appropriate choice of the time varying coefficients, this simple operation can perform a large number of functions in addition to filtering, such as estimating autocorrelations, estimating means and first order moments, estimating gradients, convolving with time variant responses, matched filtering, and discrete Fourier and other orthogonal transforms. Also by simply feeding the output back into the input a more general filtering operation is obtained. This general operation is often referred to as a recursive filter, an autoregressive filter, or an infinite impulse response filter. This basic operation as it applies to each algorithm under investigation will be developed in later sections.

In summary, the major constraint on any algorithm being investigated is that it must use this fundamental operation as the basis of its processing algorithm. This constraint will ensure fast processing, small size, low weight, low power consumption, and ruggedness.

5.1.3 Assumptions

As with any initial investigation, certain assumptions must be made about the data available at the input for processing and the type of data

needed at the output to generate the appropriate control signals. Specific assumptions about the input data imposed by the nature of the target, background, and noise will be referred to in the description of individual algorithms, since these assumptions are algorithm dependent. However, there are several general assumptions that can be made about all the algorithms.

The expected range of sampling rates of the input data must be determined. The spatial sampling of any scene is dependent on the zoom capabilities of the imager and on the number of pixels in each frame. It will be assumed that each frame of the raw data will consist of an array 100 x 100 pixels in size (although this is not a limiting assumption) with no assumptions made about the zoom capabilities of the imager. Assumptions about the time sampling are needed to specify the number of frames produced every second. This is dependent on the number of pixels that the target is allowed to move from frame to frame, which in turn is dependent on the size and velocity of the target relative to the imager and on the zoom capabilities of the imager. Since this number is target dependent, it is difficult to specify. For a target 100 feet long, extending 100 pixels in the imager output array, and moving 1000 feet per second, a frame rate of 100 frames per second would allow a movement of 10 pixels per frame. This is probably a practical upper limit with a lower limit being about 1 frame per second. Note that the upper limit corresponds to a bulk pixel rate of 1,000,000 pixels per second, which is indeed a very high data rate, but one which can be handled easily by CCDs.

The output control signals are used to keep the target within the field of view. With this feedback, the system will be expected to track a specified target. There are two approaches to deriving a set of tracking parameters. One approach is based on relative positional measurements, while the other

approach is based on absolute positional measurements. The relative approach gives parameters which indicate changes in target position, size, and angle of orientation relative to some previous estimate. The absolute approach gives parameters that specify target position, size, and angle of orientation in reference to a fixed system. The TSVIP approach is a relative one, since it gives frame to frame differences in x and y positions, differences in angle of orientation, and ratios of size changes. Note that each parameter set can be estimated from the other one, once a common reference position is established. Also note that the relative approach may allow errors in the positional estimate to accumulate, since these errors are fed back into the input of the system, while in the absolute approach these errors are not fed back. Which approach is used will be specified in the discussion of the individual algorithms in the following sections.

5.2 The Algorithms

In this section several feasible algorithms for performing image tracking are introduced. The discussion of each algorithm includes its underlying principles and the assumptions it makes about the input data. The five categories of the algorithms being investigated are referred to as gradient, moment, coordinate, transformation, and segmentation.

5.2.1 Gradient Algorithms

5.2.1.1 Development

These algorithms include the TSVIP algorithm discussed earlier in this report and Lie theoretical methods [21]. The fundamental equation of these algorithms can be developed mathematically either in terms of a Taylor series expansion of the equation governing motion of a target or in terms of group

theoretic methods, but since this development is complex and can be referenced elsewhere [5,21,22], only the results are presented here.

In theory the fundamental relation equates the time derivative of the image intensity at a point as a nonlinear function of the coordinates of the point and the spatial derivatives at that point, and is given from the Lie formulation as

$$\partial f / \partial t = a_{11}x(\partial f / \partial x) + a_{12}y(\partial f / \partial x) + a_{21}x(\partial f / \partial y) + a_{22}y(\partial f / \partial y) + b_1(\partial f / \partial x) + b_2(\partial f / \partial y) \quad (5.2.1.1)$$

where $f=f(x,y,t)$ is the image intensity as a function of the x-y image plane and time. At any given time, the six coefficients ($a_{11}, a_{12}, a_{21}, a_{22}, b_1$, and b_2) are assumed constant over all points within the target. Thus, an overconstrained set of N linear equations (where $N > 6$) in six unknowns can be derived by estimating the derivatives at N points within the target at a given point in time. These equations can then be solved for the six unknowns with any of a large number of descent procedures [16,20] that can be implemented with CCDs. Two possible solutions, the least squares pseudoinverse solution and the Widrow-Hoff sequential deterministic solution, are developed in Chapter IV of this report. One other CCD implementable solution, a Kalman filter formulation of the estimation problem, is proposed in this chapter. Since two other algorithm groups, moment and coordinate, also produce an overconstrained set of linear equations and can be solved with the same procedures, the Kalman procedure is presented in Section 5.2.4 after discussion of these algorithms.

It is interesting to compare the Lie formulation given by (5.2.1.1) with the TSVIP formulation. Even though the two equations are developed from quite different formulations, they are very similar. The only difference is that the TSVIP algorithm approximates the time derivative of the intensity

function with the difference between two consecutive frames,

$$\partial f(x, y, t) / \partial t \approx f(x, y, t+T) - f(x, y, t) \quad (5.2.1.2)$$

There is an interesting physical interpretation of the coefficients derived in these formulas, when the coefficients are constrained such that $a_{11} = a_{22} = a_1$ and $-a_{12} = a_{21} = a_2$. This constraint restricts target motion to translation, rotation, and dilation. If the target is rotating in the image plane at a rate $d\theta/dt$ about some point (x_0, y_0) and expanding at the rate of $d\rho/dt$ about this same point, which is moving with rectangular velocity components given by dx/dt and dy/dt , then the constrained parameters give the following relationships:

$$d\theta/dt = \tan^{-1}(a_2/a_1) \quad (5.2.1.3a)$$

$$1 + (1/\rho) d\rho/dt = \sqrt{a_1^2 + a_2^2} \quad (5.2.1.3b)$$

$$dx/dt = b_1 \quad (5.2.1.3c)$$

and

$$dy/dt = b_2 \quad (5.2.1.3d)$$

If for some small time interval, T , these derivatives ($d\theta/dt$, $d\rho/dt$, dx/dt , and dy/dt) can be assumed constant, then a constrained Affine transformation of coordinates can be given by

$$\begin{bmatrix} x(t+T) \\ y(t+T) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (5.2.1.4)$$

where

$$a_1 = [\rho(t+T)/\rho(t)] \cos[\theta(t+T) - \theta(t)]$$

$$a_2 = [\rho(t+T)/\rho(t)] \sin[\theta(t+T) - \theta(t)]$$

$$b_1 = x_0(t+T) - x_0(t)$$

and

$$b_2 = y_0(t+T) - y_0(t)$$

Thus, a relativistic set of parameters about changes in target orientation, position, and size between consecutive frames can be generated as:

$$\Delta x = b_1 \quad (5.2.1.5a)$$

$$\Delta y = b_2 \quad (5.2.1.5b)$$

$$\Delta \theta = \tan^{-1} (a_2/a_1) \quad (5.2.1.5c)$$

and

$$\alpha = \sqrt{a_1^2 + a_2^2} \quad (5.2.1.5d)$$

5.2.1.2 Assumptions

There are several major assumptions made by gradient algorithms about the type of data available for processing. One important assumption is that the target has been roughly segmented in some way from the background, so that a set of known target points exists for processing. Also the target must be large enough so that a sufficient number of sampling points exist. Note that no assumptions are necessary for the background points. In fact a background is not even required, unless it is needed for segmentation.

Another important assumption is that the spatial and time derivatives at a point can be easily estimated from the sampled intensity values at that point and its surrounding points. This implies that the spatial and time sampling rates must be high enough for good estimates, but not so high that the equations are ill-conditioned. This requirement also relates to the texture and motion of the target in that variations in image intensity values from sampled point to sampled point can neither be too large nor too small. Also since derivative estimates are very sensitive to noise, the noise levels need to be low. This can be achieved to some degree by low pass filtering the input data before the derivative estimates are made.

A third assumption is that an additional algorithm is being used to provide absolute estimates of the target position. This is necessary since

relative estimates can accumulate error and, without an additional algorithm, can lose track of the target.

The final assumption is that the target is restricted to motion as given by the unconstrained Affine transform. Since most physical motions will tend to be ones of this type, this assumption is not too critical.

5.2.2 Moment Algorithms

5.2.2.1 Development

These algorithms are based on means and moments of the input data as opposed to gradients. They perform an averaging of the data, and thus are less sensitive to noise and sampling rates than gradient based algorithms. However, they are very similar to the gradient algorithms in that they establish a fundamental equation at each point in a target describing its spatial and time perturbations. This fundamental equation is based on the same six unknowns and is derived from the fundamental gradient equation by a spatial integration of it. The integration is carried over a fixed area centered on a target point. If the area is defined by $a \leq x \leq b$ and $c \leq y \leq d$, then the integration operation on the fundamental gradient equation yields:

$$\begin{aligned} \int_c^d \int_a^b (\partial f / \partial t) dx dy &= a_{11} \int_c^d \int_a^b x (\partial f / \partial x) dx dy + a_{12} \int_c^d \int_a^b y (\partial f / \partial x) dx dy \\ &+ b_1 \int_c^d \int_a^b (\partial f / \partial x) dx dy + a_{22} \int_a^b \int_c^d y (\partial f / \partial y) dy dx \\ &+ a_{21} \int_a^b \int_c^d x (\partial f / \partial y) dy dx + b_2 \int_a^b \int_c^d (\partial f / \partial y) dy dx \end{aligned} \quad (5.2.2.1)$$

This equation can be further reduced by noting the following equalities of calculus:

$$\int_c^d \int_a^b [\partial f(x,y,t)/\partial t] dx dy = \partial V(t)/\partial t \quad (5.2.2.2a)$$

$$\int_c^d \int_a^b x [\partial f(x,y,t)/\partial x] dx dy = b \int_c^d f(b,y,t) dy - a \int_c^d f(a,y,t) dy - V(t) \quad (5.2.2.2b)$$

$$\int_a^b \int_c^d y [\partial f(x,y,t)/\partial y] dy dx = d \int_a^b f(x,d,t) dx - c \int_a^b f(x,c,t) dx - V(t) \quad (5.2.2.2c)$$

$$\int_c^d \int_a^b y [\partial f(x,y,t)/\partial x] dx dy = \int_c^d y f(b,y,t) dy - \int_c^d y f(a,y,t) dy \quad (5.2.2.2d)$$

$$\int_a^b \int_c^d x [\partial f(x,y,t)/\partial y] dy dx = \int_a^b x f(x,d,t) dx - \int_a^b x f(x,c,t) dx \quad (5.2.2.2e)$$

$$\int_c^d \int_a^b [\partial f(x,y,t)/\partial x] dx dy = \int_c^d f(b,y,t) dy - \int_c^d f(a,y,t) dy \quad (5.2.2.2f)$$

$$\int_a^b \int_c^d [\partial f(x,y,t)/\partial y] dy dx = \int_a^b f(x,d,t) dx - \int_a^b f(x,c,t) dx \quad (5.2.2.2g)$$

where

$$V(t) = \int_c^d \int_a^b f(x,y,t) dx dy$$

Note that this reduction eliminates all but one derivative operation, and replaces them with zero and first order moment calculations of the windowed area and its boundaries. Again the single time derivative operation in (5.2.2.2a) can be replaced by

$$\partial V(t)/\partial t \approx V(t+T) - V(t) \quad (5.2.2.3)$$

as long as the time sampling rate is fast enough or the image evolves slow enough to make this approximation valid.

Since the data are discrete, the integral operations must also be approximated by some numerical integration technique. The simplest and most easily implementable with CCDs is the rectangular rule:

$$\int_a^b g(x) dx = \sum_{n=0}^{N-1} g(a+n\Delta x) \Delta x \quad (5.2.2.4)$$

where $\Delta x = (b-a)/N$. Thus, all the integration operations in (5.2.2.2) can be developed as weighted sums of ordered pixels and thereby be implemented with CCD based structures.

Since the coefficients are identical in the two formulations, they have the same interpretation as given by (5.2.1.3), and they generate the same relative set of control signals as given by (5.2.1.5). Also they can be calculated using the pseudo-inverse formulation or the Widrow-Hoff formulation as presented earlier in this report, or using the Kalman formulation presented in Section 5.2.4.

5.2.2.2 Assumptions

With few exceptions the assumptions made by these algorithms are essentially identical to those made by the gradient based algorithms as presented in Section 5.2.1.2. However these algorithms do not require special constraints on the spatial sampling rate or the texture of the object, nor do they require a low noise level as in the gradient algorithms.

5.2.3 Coordinate Algorithms

5.2.3.1 Derivation

The coordinate based algorithms also produce an overdetermined set of linear equations as do the gradient and moment algorithms, but they do not require the estimation of either the gradient at any point or the moment of any region. They do however require the accurate identification of the locations of a fixed number of points in consecutive frames. If a subset of N points from the target can be accurately tracked in each frame, then the target as a whole can also be tracked. Labelling these N points as

$(x_1(t), y_1(t)), (x_2(t), y_2(t)), \dots, (x_N(t), y_N(t))$, the linear set of equations become

$$\begin{aligned}
 x_1(t+T) &= b_1 + a_{11}x_1(t) + a_{12}y_1(t) \\
 y_1(t+T) &= b_2 + a_{21}x_1(t) + a_{22}y_1(t) \\
 x_2(t+T) &= b_1 + a_{11}x_2(t) + a_{12}y_2(t) \\
 y_2(t+T) &= b_2 + a_{21}x_2(t) + a_{22}y_2(t) \\
 &\vdots \\
 x_N(t+T) &= b_1 + a_{11}x_N(t) + a_{12}y_N(t) \\
 y_N(t+T) &= b_2 + a_{21}x_N(t) + a_{22}y_N(t)
 \end{aligned} \tag{5.2.3.1}$$

The coefficients correspond to those of the unconstrained Affine transformation as given in (5.2.1.1) or (5.2.2.1), and thus have the same physical interpretation and output the same control signals as both the gradient and moment algorithms. They can also be solved using the same methods.

The major difference is that accurate segmentation of a few points is required. If that can be done, and since this formulation does not need the image intensity value at any point, both noise problems and spatial and time sampling constraints can be ignored. Since noise does not work into the equations, a more accurate solution can be obtained. Unfortunately, the task of tracking a select set of image points is very target dependent and may be too difficult to implement with any reasonable algorithm or hardware.

5.2.3.2 Assumptions

The coordinate based algorithms have only two major assumptions. The first and most prohibitive is that accurate positional identification of a select set of points is required in successive frames. This assumes that

the target consists of identifiable points such as the intersections of edges or boundaries, points that are consistently the largest or smallest values from scene to scene, or points that come from matched filtering with regions of known size and shape. It is this assumption that may prohibit coordinate based algorithms from being implementable.

Another assumption is that target motion is restricted to those allowed by the unconstrained Affine transform. Motions such as translation, dilation, and rotation within the x-y viewing plane are nicely modeled by the Affine parameters, but rotations about an axis that is not orthogonal to the viewing plane are not. Even so, this restriction is essentially a mild one.

5.2.4 A Kalman Based Solution of the Affine Parameters

In this section a solution technique for solving an overdetermined set of equations in six unknowns is presented. This technique applies to the gradient, moment, and coordinate generated equations. In fact, since the coefficients are identical in all three approaches, a novel idea might be to use N gradient derived equations, M moment derived equations, and P coordinate derived equations as the input data for either this Kalman technique or any of the others discussed in this paper. Care should be taken, however, so that the data from all three sources are about the same order of magnitude, else the equations would be ill-conditioned. Normalization can be easily handled by adjusting the CCD tap weights as this data is computed.

The development of this technique is presented in vector form, so let

$$A^T(k) = [a_{11}(k) \ a_{12}(k) \ a_{21}(k) \ a_{22}(k) \ b_1(k) \ b_2(k)] \quad (5.2.4.1a)$$

and

$$x^T(k) = [x_1(k) \ x_2(k) \ x_3(k) \ x_4(k) \ x_5(k) \ x_6(k)] \quad (5.2.4.1b)$$

where k is the iteration index, $A(k)$ represents the sequential estimates of the unconstrained Affine parameters, and $X^T(k)$ represents the k^{th} data set derived either by gradient, moment, or coordinate calculations. The elements of $X^T(k)$ represent the data weights on the corresponding coefficients in $A(k)$ so that the k^{th} derived equation from any of the techniques is represented by

$$Z(k) = A^T(k)X(k) \quad (5.2.4.2)$$

where $Z(k)$ is the left side quantity as given in (5.2.1.1), (5.2.2.1), or (5.2.3.1).

Since the Affine parameters are assumed constant over all observations in a particular frame, $A(k)$ can be modeled as the following Markov process:

$$A(k+1) = A(k) + u(k) \quad (5.2.4.3)$$

where $u(k)$ is modeled as a stationary white noise process with variance V_u , thus allowing for the error in the parameter estimates from different observations. The observation equation can then be given as

$$Z(k) = A^T(k)X(k) + v(k) \quad (5.2.4.4)$$

where $v(k)$ is modeled as a stationary white noise process with variance V_v . Assume that $A(0)$ is a random vector with variance $V_A(0)$.

Finally the estimation equations come directly from applying the above equations to Kalman filter theory, and are given as

$$A(k+1) = A(k) + \rho(k)X(k)[Z(k) - A^T(k)X(k)] \quad (5.2.4.5a)$$

$$\rho(k) = V_A(k)/[V_v^2 + X^T(k)V_A(k)X(k)] \quad (5.2.4.5b)$$

and

$$V_A(k+1) = [I - \rho(k)X(k)X^T(k)]V_A(k) + V_u \quad (5.2.4.5c)$$

Thus, all that is needed to start the sequence of estimation is to choose V_u , V_v , $A(0)$, and $V_A(0)$. The rate of convergence is influenced by both V_u and V_v .

If V_v is chosen to be small, then from (5.2.4.5c) the product $(\rho(k)X(k)X^T(k))$ is nearly equal to the identity matrix and $V_A(k)$ will converge to its minimum value V_u too quickly. Conversely if V_v is chosen to be large, $V_A(k)$ will tend to V_u very slowly. The choices for $A(0)$ and $V_A(0)$ are interdependent, in that $V_A(0)$ should be chosen small if the initial estimate $A(0)$ is thought to be accurate or $V_A(0)$ should be chosen large if the estimate is believed to be inaccurate. Note that the estimates should vary little from frame to frame, so that if the estimate from the previous frame is used for $A(0)$, then $V_A(0)$, V_v , and V_u should be chosen small for quick convergence to an accurate solution. A few startup frames should be sufficient to accurately choose $A(0)$.

Implementation of this technique is more difficult than the Widrow-Hoff technique, but note the estimation similarities. The major difference is that the gain on the error term is a 6×6 matrix in this formulation, and it is a scalar in the Widrow-Hoff formulation. In spite of the higher complexity, the Kalman formulation is still CCD implementable since it does not require any large inverses but rather parallel computation of inner products. Also the increase in hardware over the Widrow-Hoff estimator may be an acceptable trade-off for the decrease in convergence time offered by the Kalman estimator.

5.2.5 Transformation Algorithms

5.2.5.1 Derivation

The basic principle behind these algorithms is to transform the image input data into a new set of data where the computations of rotational, translational, and dilational parameters are made easier. This section investigates some of the properties of the Fourier transform and a geometric

coordinate transform denoted as the Log-Polar transform. The function of each of these transforms in computing certain motion parameters is then discussed.

The spatial Fourier transform of an image $f(x,y,t)$ is given by

$$F(u,v,t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y,t) \exp[-2\pi j(ux+vy)] dx dy \quad (5.2.5.1)$$

The following properties show the effects that translation, dilation, and rotation of an image $f(x,y,t)$ has upon its Fourier transform. If $f(x,y,t) \leftrightarrow F(u,v,t)$ indicates the Fourier pair, then

$$f(x-x_0, y-y_0, t) \leftrightarrow \exp[-2\pi j(ux_0 + vy_0)] F(u,v,t) \quad (5.2.5.2a)$$

$$f(ax, ay, t) \leftrightarrow (1/a)^2 [F(u/a, v/a)] \quad (5.2.5.2b)$$

and

$$f(r, \theta+\theta_0) \leftrightarrow F(w, \phi+\theta_0) \quad (5.2.5.2c)$$

where $f(r, \theta+\theta_0)$ represents a θ_0 rotation of the image about the origin.

There are three important results of the above properties. The first is that the magnitude of the Fourier transform is invariant to translation. The second is that scaling the image function also scales the Fourier transform. And lastly, a rotation about the origin in the image plane results in a rotation about the origin of the same amount in the Fourier transform plane.

The Log-Polar transform is based on the conversion from rectangular to polar coordinates and is given as

$$L(u,v,t) = f[\ln(x^2 + y^2)^{1/2}, \tan^{-1}(y/x)] \quad (5.2.5.3)$$

Basically, the image function is converted to polar coordinates, the natural logarithm is applied to its magnitude coordinate, and the new coordinate system is interpreted as rectangular coordinates. If $f(x,y,t) \rightarrow L(u,v,t)$ represents this transformation then the following properties result:

$$f(ax, ay, t) \rightarrow L(u + \ln(a), v, t) \quad (5.2.5.4a)$$

and

$$f(r, \theta + \theta_0) \rightarrow L(u, v + \theta_0, t) \quad (5.2.5.4b)$$

where $f(r, \theta + \theta_0)$ represents a θ_0 rotation of the image about the origin. Note that this transformation produces a simple translation in the transformed domain given a rotation and dilation in the image domain. Also, using the translation invariance of the magnitude of the Fourier transform, a transformation of the image function that is invariant to translation, rotation, and dilation can be obtained. This is done by computing the magnitude of the Fourier transform of the Log-Polar transform of the magnitude of the Fourier transform of the original data. This invariant transform is useful in pattern recognition problems by characterizing objects subject to rotation, dilation, and translation. If the image loses track of the target, a pattern recognition routine can take over to match the last frame containing the target with a new frame zoomed back to show a much larger image plane, and this transform can be used to locate the missing target.

A transform of the image plane which is invariant to translation, rotation, and dilation is certainly useful, but doesn't yield the parameters that indicate the amount of movement. To do this using these transforms, matched filtering is required. The actual procedure that must be performed is too complex to detail in this report, so only a brief description of this procedure is outlined.

Matched filtering is essentially a normalized cross-correlation between a function and a reference function. In this application, a windowed region containing part of the target of a previous frame is used as a reference function and is matched filtered with the current frame where the target

has undergone only translation. The matched filter produces an image that has a value at the location where the target has shifted to. Unfortunately the target is only allowed to undergo translation.

If the target undergoes translation, rotation, and dilation, then the rotation and dilation parameters can be found by match filtering the Log-Polar transform of the magnitude of the Fourier transform of the reference plane with a similar transformation of the current plane. The peak value in the match filter output indicates the amount of rotation and dilation.

Once these coefficients are found, they can be applied to the current image to correct for any rotation and dilation of the reference image. The result can then be inverse Fourier transformed and match filtered with the reference image. The peak now indicates the translation movements since the reference image.

The number of computations that must be performed to derive the movement parameters is enormous. However, many of the necessary operations can take advantage of the high speed and parallel processing nature of CCDs. The Fourier transform is implemented with the discrete Fourier transform which has a CCD implementable form. Also match filtering is simply convolution with a spatially reversed reference, and CCDs can be structured to handle two dimensional convolutions. Operations such as tangents, logarithms, magnitudes, and squaring can be implemented with special purpose high speed analog or digital hardware.

5.2.5.2 Assumptions

There are several assumptions made by this procedure for identifying the tracking parameters. The foremost one is that this algorithm allows only translation, dilation, and rotation. Noise and rotation about an axis not perpendicular to the viewing plane are not accounted for, and thus will

cause errors in the parameters that are measured. Another assumption is that the noise must either be low level or stationary and white, since the match filter is optimum only in the presence of white noise. Finally, unlike the previous algorithms, large changes in position, size, and orientation are all allowed by this algorithm, and so no assumptions about the time sampling rate need to be made.

5.2.6 Segmentation Algorithms

The purpose of these algorithms is to segment the target from the background and then use this segmented image to estimate the target size, position, and orientation. There are two computational parts to these algorithms: Segmentation and parameter estimation. Three segmentation methods denoted by intensity methods, texture methods, and Gestalt methods are suggested. Two parameter estimation methods denoted by principal axis methods and projection methods are proposed.

5.2.6.1 Intensity Segmentation

These methods assume that the gray levels belonging to the target pixels are significantly different from those of the background. In this case segmentation can be achieved by thresholding with the appropriate thresholds. The difficulty lies in the determination of these thresholds. If some a priori knowledge is available about the distribution of target and background intensity values is available, then a Bayesian criterion can be used to determine the optimum thresholds. If not, then a histogram of the image can be computed and the intensity levels belonging to the target can be identified if only the shape of the target distribution is known. Another possibility would be to apply the above methods to either a low-pass, high-pass, or band-pass version of the image. The filtered image may have target pixels whose intensity are distinct from those of the background.

The exact method used for intensity segmentation is highly dependent on the assumptions made about the target. Since the expected range of applications is large, no assumptions will be presented in this investigation. Only possible procedures are suggested.

5.2.6.2 Texture Segmentation

These algorithms assume that the texture of the target is significantly different from that of the background. Texture, however, is a subjective quality. One means of giving texture a quantitative measure comes from linear prediction theory. Linear prediction provides a set of coefficients which will vary from texture to texture, and is based on linearly predicting the intensity value of a pixel by a weighted sum of surrounding pixels. These weights are the linear prediction coefficients, are assumed to be constant over a given texture, and can be computed by structuring an over-constrained set of linear equations where the coefficients are the unknowns, and using any of the CCD implementable techniques suggested in this report for solving this kind of problem. Once the weights for a particular type of texture are found, then the linear prediction can be applied to the image array as a CCD implementable convolution. The squared error between the predicted value and actual value can then be thresholded to determine whether or not a particular pixel belongs to the texture on which the coefficients were trained. Thus the segmentation is complete. Results of this approach to image segmentation are given in [23]. A filtered version of the image may provide a better texture with which to work, but this is dependent on any assumptions made a priori about the target.

A major assumption made by this algorithm is that some kind of rough segmentation is required to identify the pixels to be used in training the linear predictor coefficients. If the target is expected to be in the

center of the image (as during tracking), then the texture filter can be trained there or along the edges.

5.2.6.3 Gestalt Segmentation

In this approach the segmentation method is based on the Gestalt law of common fate, which implies that if several image regions appear to move together, they are treated as a single object. The algorithms used in this technique may be the same as those used for gradient, moment, coordinate, or transform algorithms discussed in earlier sections. Here these algorithms are applied to small nonoverlapping regions within the input image and the movement parameters of each region are computed. Those regions whose parameters are close in some vector norm sense are grouped either to the target or background, thus performing segmentation.

This technique makes the same assumptions as the tracking algorithm being used. The primary difference is that the regions are smaller and fewer points are available for computing the tracking parameters within each region. Also this type of segmentation is rougher, since it assigns regions of pixels to target or background as opposed to individual pixels.

5.2.6.4 Principal Axis Parameter Estimation

Or a binary image is produced from the appropriate segmentation algorithm, an estimate of target position, orientation, and size is needed. The principal axis method first estimates the centroid of the target image by summing the coordinate values of points belonging to the target. The average x and y coordinate that results is used as the position of the target. Next a line which contains that point is obtained by minimizing the sum of the square of the perpendicular distances from all target points to that line with respect to the slope of the line. The angle formed between the

line (principal axis) and a reference line yields the angle of orientation. The size estimate is then made by computing the number of pixels classified as belonging to the target. Note that the above calculations can be done with weighted sums of coordinate values, and thus can be implemented with CCDs.

5.2.6.5 Projection Parameter Estimation

A faster algorithm may be provided by this approach. This method first projects the binary image onto the x and y axes. That is, the number of pixels belonging to the target along each row and along each column is computed and stored. As in the principal axis approach the total number of target classified pixels is used as the size estimate. Now however, the position is determined by computing the one-dimensional centroids of the two orthogonal projections. The one-dimensional centroid can be computed by a summation of coordinate values weighted by the number of pixels from the target assigned to that coordinate. The angle of orientation can be computed by performing this one-dimensional centroid calculation on the four sections of the projections separated at the centroids computed earlier. The angle of orientation is thus given by

$$\theta = \tan^{-1} [(Y^T - Y^B) / (X^T - X^B)] \quad (5.2.6.1)$$

where Y^T , Y^B , X^T , and X^B are the centroids for the top and bottom y projection and the top and bottom x projection respectively. Note that the computations made by this algorithm can be performed mostly by CCDs.

5.2.6.6 Assumptions

There are several assumptions made by segmentation algorithms. One important assumption is that the entire target is in the field of view of the input image array. This is necessary so that the differentiation between

target and background can be used for segmentation and position estimation. Another important assumption is that there is a difference between the target and background, and that the difference is dependent on the application and therefore must be known a priori. This is so the hardware can be structured to segment based on only one type of difference, whether that be intensity differences, texture differences, or motion differences. Although the parameter estimation algorithms are relatively insensitive to a small number of misclassifications, the algorithms will not work well with only a rough segmentation performed.

5.3 Implementation

As a demonstration of the practicality of this algorithm search, an example system is developed in this section. This example is not a proposal for hardware design, nor does it necessarily represent the optimum approach, but it does serve to illustrate the CCD structure of a viable alternative to the proposed system. The design presented in this section is based on the moment tracking algorithm of section 5.2.2 and the Kalman parameter identification algorithm of section 5.2.4. It is assumed that there is a single target centered in the field of view (FOV) occupying a significant area (greater than 1% of FOV). The problem of identifying and locking the target within the field of view is not considered in this design. The overall system configuration is shown in Figure 5-1. The functions of the imager, preprocessor, moment estimator, parameter estimator, and imager controller are discussed individually in the follow sections. In addition, functional diagrams of the moment and parameter estimators are developed. The timing and control functions are discussed within each of the other functional block descriptions.

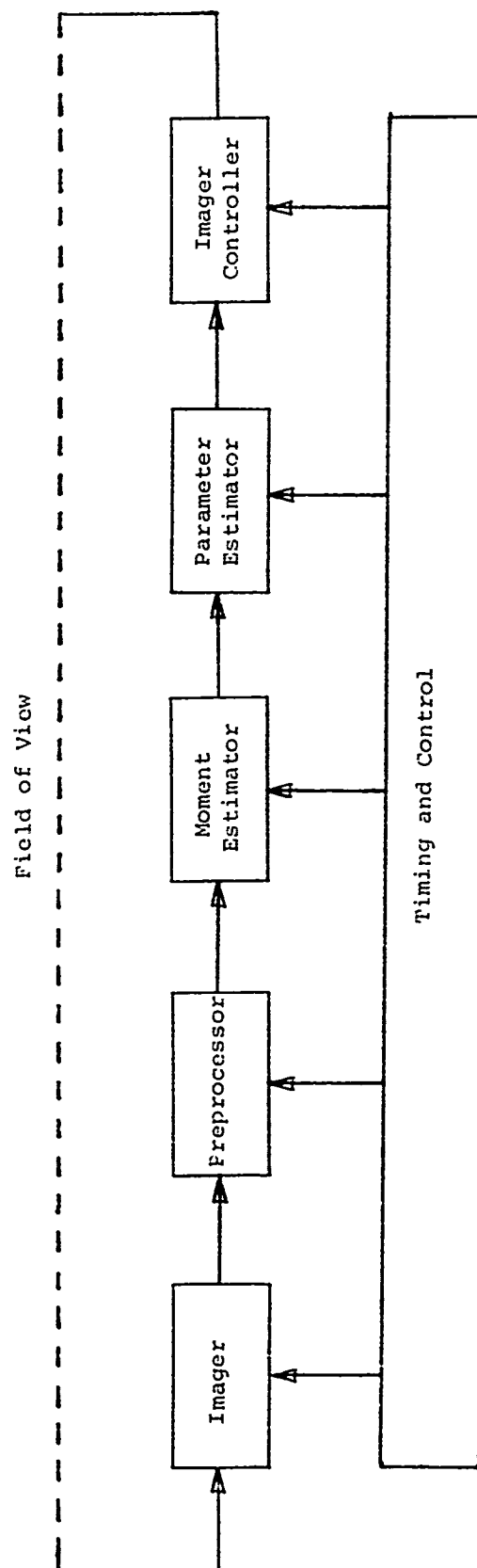


Figure 5.1
Example System Configuration

5.3.1 Imager

The imager is a CCD image array of 100 by 100 pixels. It is highly sensitive to the infrared spectrum, since it is assumed that all targets of interest will appear warm against a cold background. The exposure (integration) time is variable and controlled by the timing and control unit. The amount of exposure is determined by the light intensity of the field of view, so that the contrast between the target and background is sharp enough for accurate identification of target points by their intensity values. A frame rate of 10 frames per second allows a maximum exposure time of .01 seconds and a processing time of .09 seconds before the next frame is exposed. After exposure, the discrete analog data is transferred from the CCD image array one pixel at a time at a 1 MHz rate. A gated 10 KHz clock is used to clock the data up one row with the top row being lost and the bottom row given appropriate values in preparation for the next exposure. The top row is clocked to the right with a 1 MHz clock with a readout on the upper right corner pixel. Thus the data is clocked out with the appropriate signals from the timing and control box in the order shown in Figure 5-2. Note that if the top N rows are not needed, then they may be clocked out at a rapid rate, and readout may start on the $N + 1$ row.

5.3.2 Preprocessor

The primary function of the preprocessor section is to prepare the data for the moment estimator. This may include low pass filtering to eliminate high frequency noise or weighted correction of nonlinearities in the imager. In this design, it is used to normalize the data, so that the target contrasts sharply with the background and there are sufficient intensity variations within the target to provide good moment estimates. Histogram equalization is one means of performing this normalization, but the associated hardware

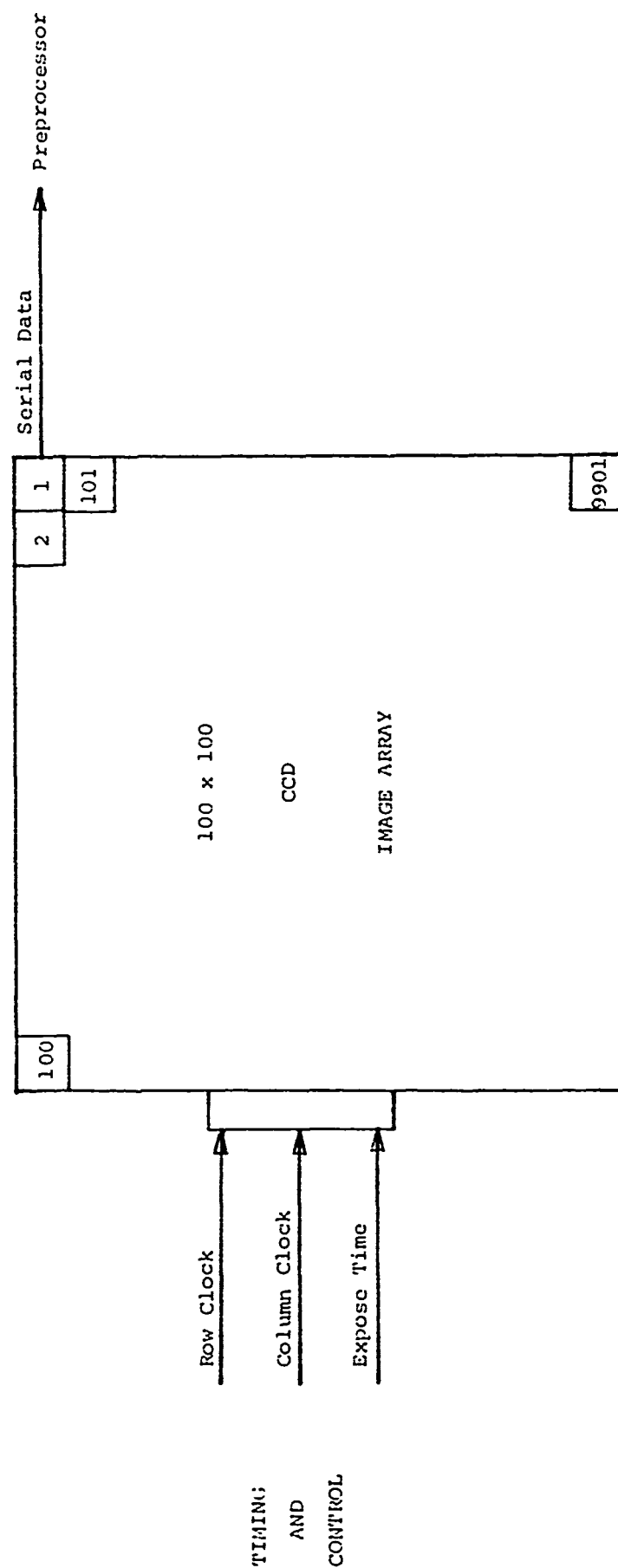


Figure 5.2
Imager With Data Readout Sequence

is too complex. A simpler but effective method is to subtract out the mean of the data at each point, then divide the resulting data by its variance. This will give a normalized data set with zero mean and unit variance.

Another function of this preprocessor is to reduce the bulk of the data. Since the target is assumed to be roughly in the center of the frame, only the center 32 by 32 pixels are kept for processing. Since the timing and control unit knows when each pixel is clocked out of the image array, it can generate the appropriate signals for clocking the center area into a 1024 stage CCD shift register. A diagram of the preprocessor with the data reduction and normalization functions is shown in Figure 5-3.

5.3.3 Moment Estimator

The moment estimator receives the data from the preprocessor and stores it in a 100 stage CCD array, which is tapped in such a way that 4 by 4 square regions are available for moment estimation. These sixteen taps are then multiplied by their digital coordinates and summed to provide the five moment estimates needed for the constrained Affine parameter estimation. Let Figure 5-4 establish a coordinate system. Using the expansion given by (5.2.2.2) and the approximation given by (5.2.2.3) and (5.2.2.4), and constraining the Affine parameters such that $a_{11} = a_{22} = a_1$ and $-a_{12} = a_{21} = a_2$, the basic equation of (5.2.2.1) becomes

$$w(k, t+T) = w(k, t) = a_1 x_1(k, t) + a_2 x_2(k, t) + b_1 x_3(k, t) + b_2 x_4(k, t) \quad (5.3.3a)$$

where

$$w(k, t) = \sum_{i=0}^3 \sum_{j=0}^3 f(x+i, y+j, t) \quad (5.3.3b)$$

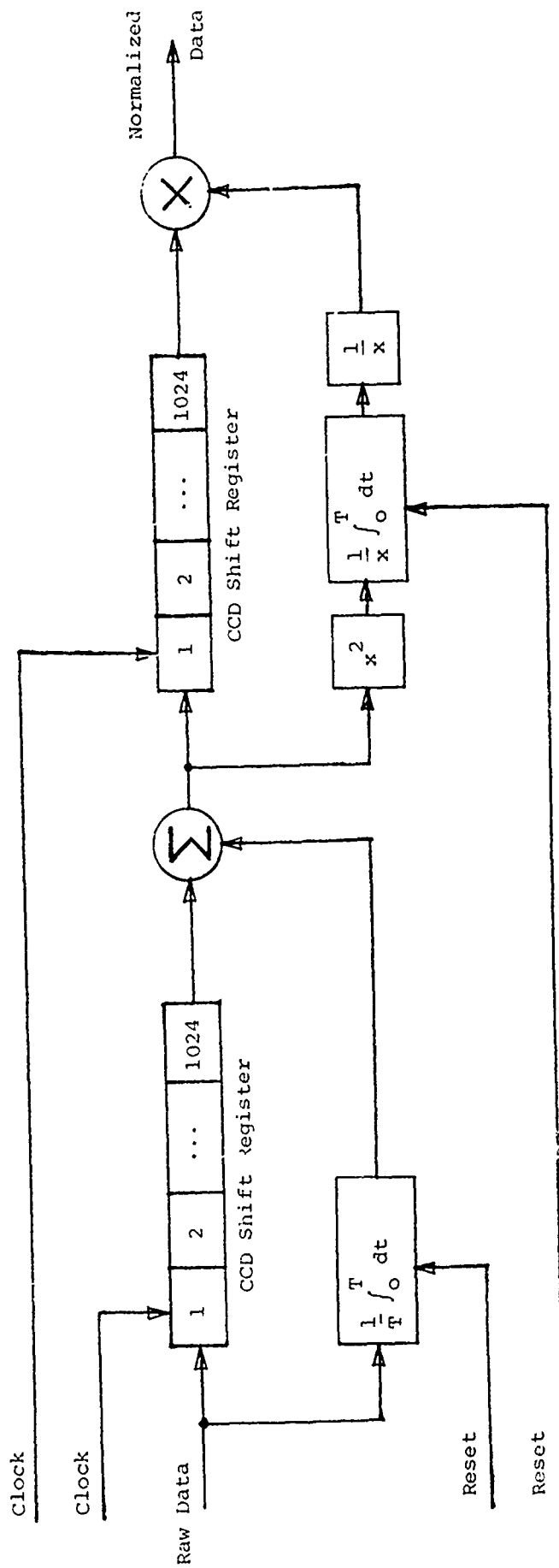


Figure 5.3
Preprocessor with normalization and storage

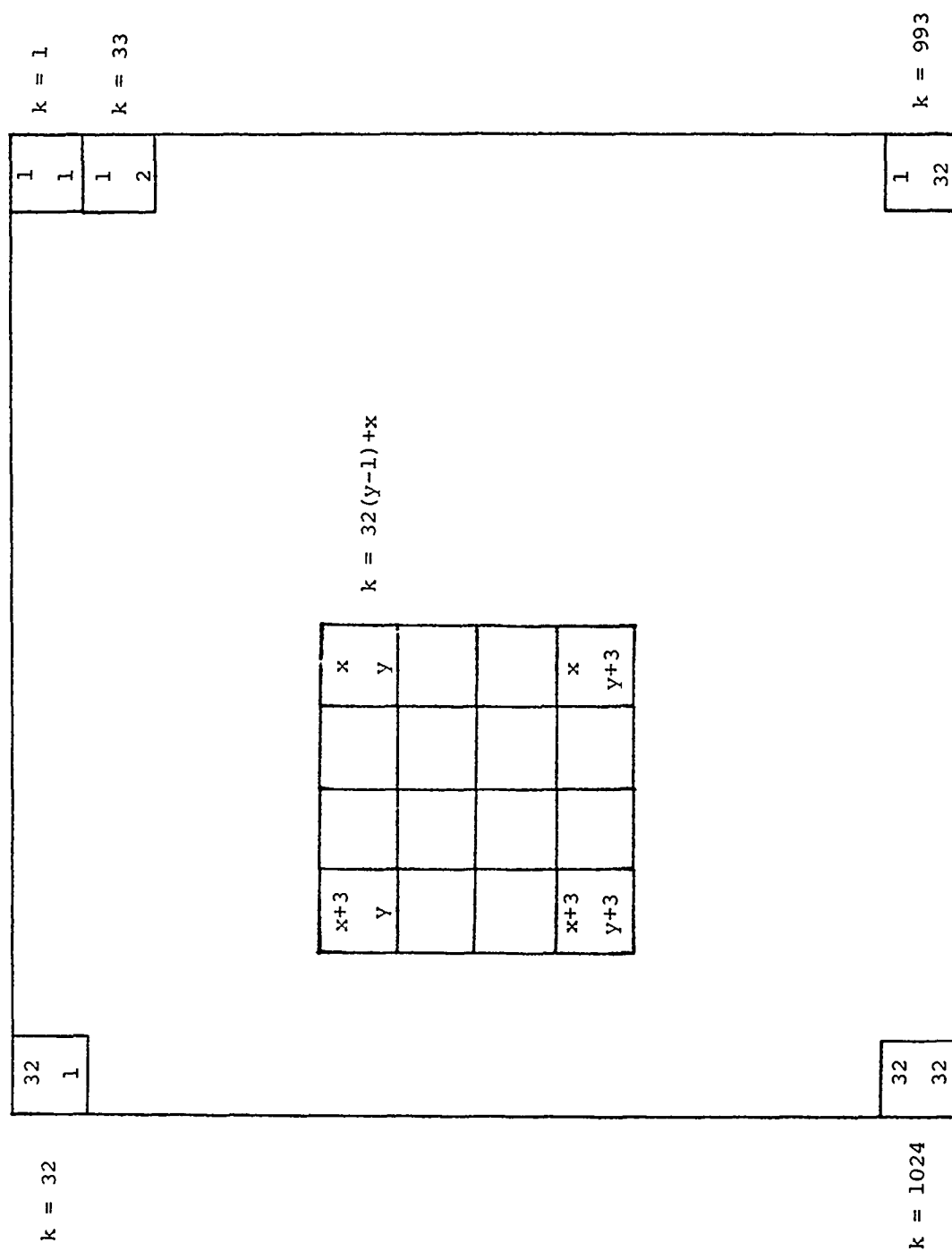


Figure 5.4
Relationship between coordinates and iteration index, k ,
of a 32 by 32 subimage

$$\begin{aligned}
x_1(k,t) = & \sum_{i=0}^3 [(y+3)f(x+i, y+3, t) - (y)f(x+i, y, t)] \\
& + \sum_{j=0}^3 [(x+3)f(x+3, y+j, t) - (x)f(x, y+j, t)] \\
& - 2 \sum_{i=0}^3 \sum_{j=0}^3 f(x+i, y+j, t)
\end{aligned} \tag{5.3.3c}$$

$$\begin{aligned}
x_2(k,t) = & \sum_{i=0}^3 (x+i) [f(x+i, y+3, t) - f(x+i, y, t)] \\
& - \sum_{j=0}^3 (y+j) [f(x+3, y+j, t) - f(x, y+j, t)]
\end{aligned} \tag{5.3.3d}$$

$$x_3(k,t) = \sum_{j=0}^3 [f(x+3, y+j, t) - f(x, y+j, t)] \tag{5.3.3e}$$

and

$$x_4(k,t) = \sum_{i=0}^3 [f(x+i, y+3, t) - f(x+i, y, t)] \tag{5.3.3f}$$

Note that these sums of products are CCD implementable. The coordinates are digital values supplied by the timing and control unit, thus every multiplication is analog by digital with analog output and every summation is analog. The diagram of the hardware needed for this moment estimator is shown in Figure 5-5. Figures 5-6 and 5-7 show further details of the multiplication and summation sections.

5.3.4 Parameter Estimator

The parameter estimator implements the Kalman estimator as given in (5.2.4.5) and is formulated for this problem as:

$$A(k+1) = A(k) + V_A(k) X(k) [Z(k) - A^T(k) X(k)] / \rho(k) \tag{5.3.4a}$$

$$V_A(k+1) = V_A(k) + V_u - V_A(k) X(k) X^T(k) V_A(k) / \rho(k) \tag{5.3.4b}$$

where

$$\rho(k) = V_V^2 + X^T(k) V_A(k) X(k) \tag{5.3.4c}$$

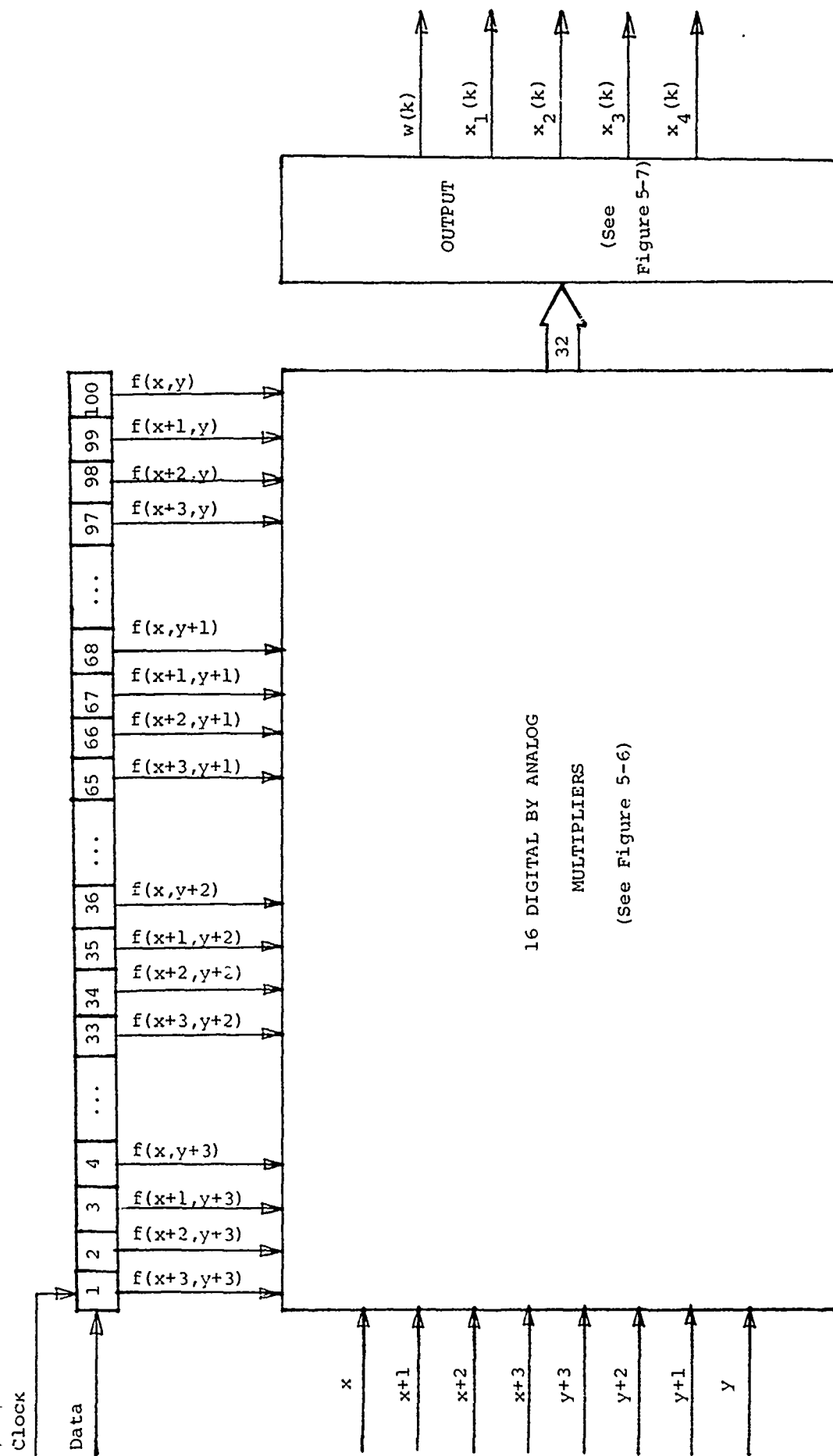


Figure 5.5
Moment Estimator

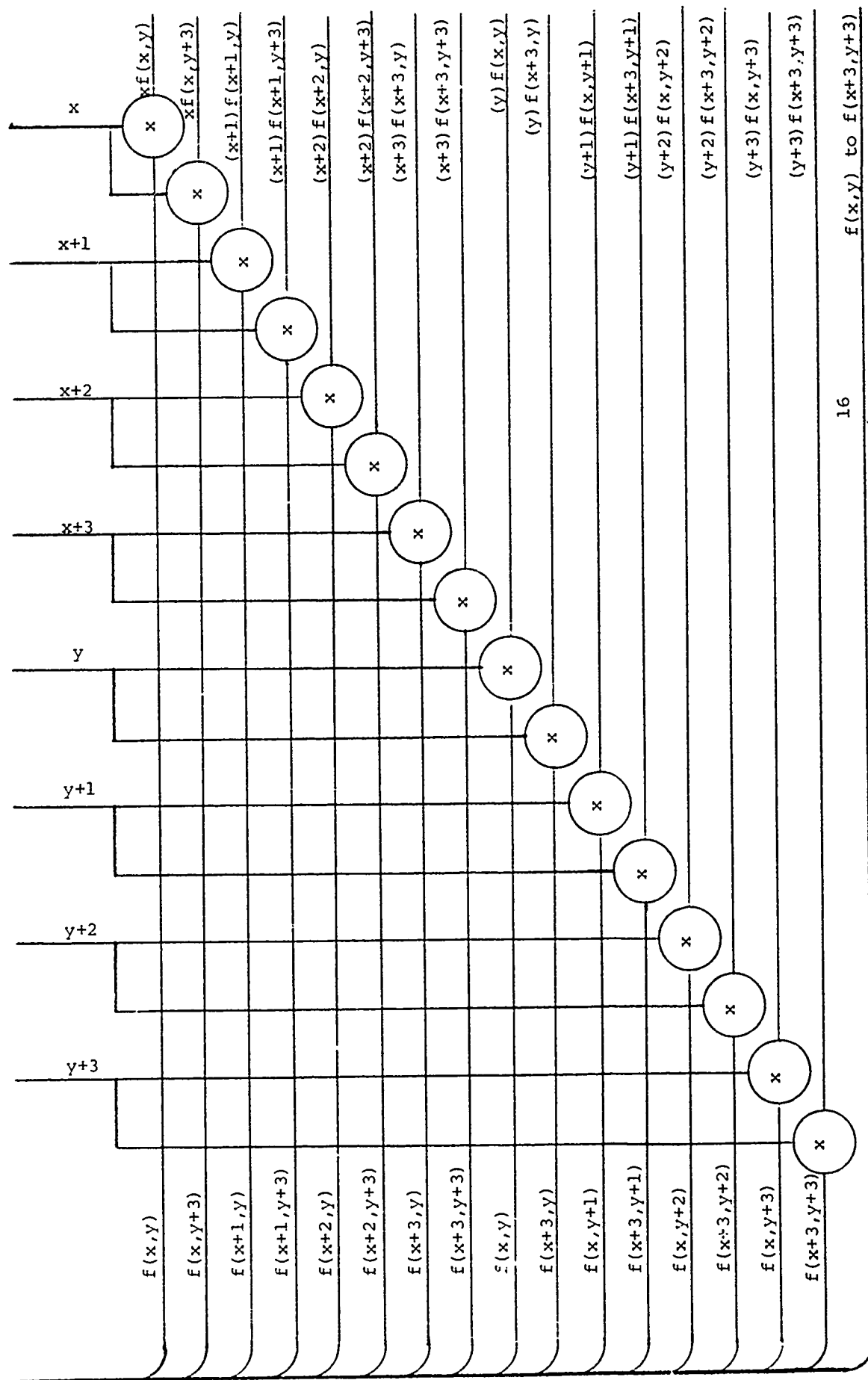


Figure 5.6
Digital by analog multipliers for moment estimator

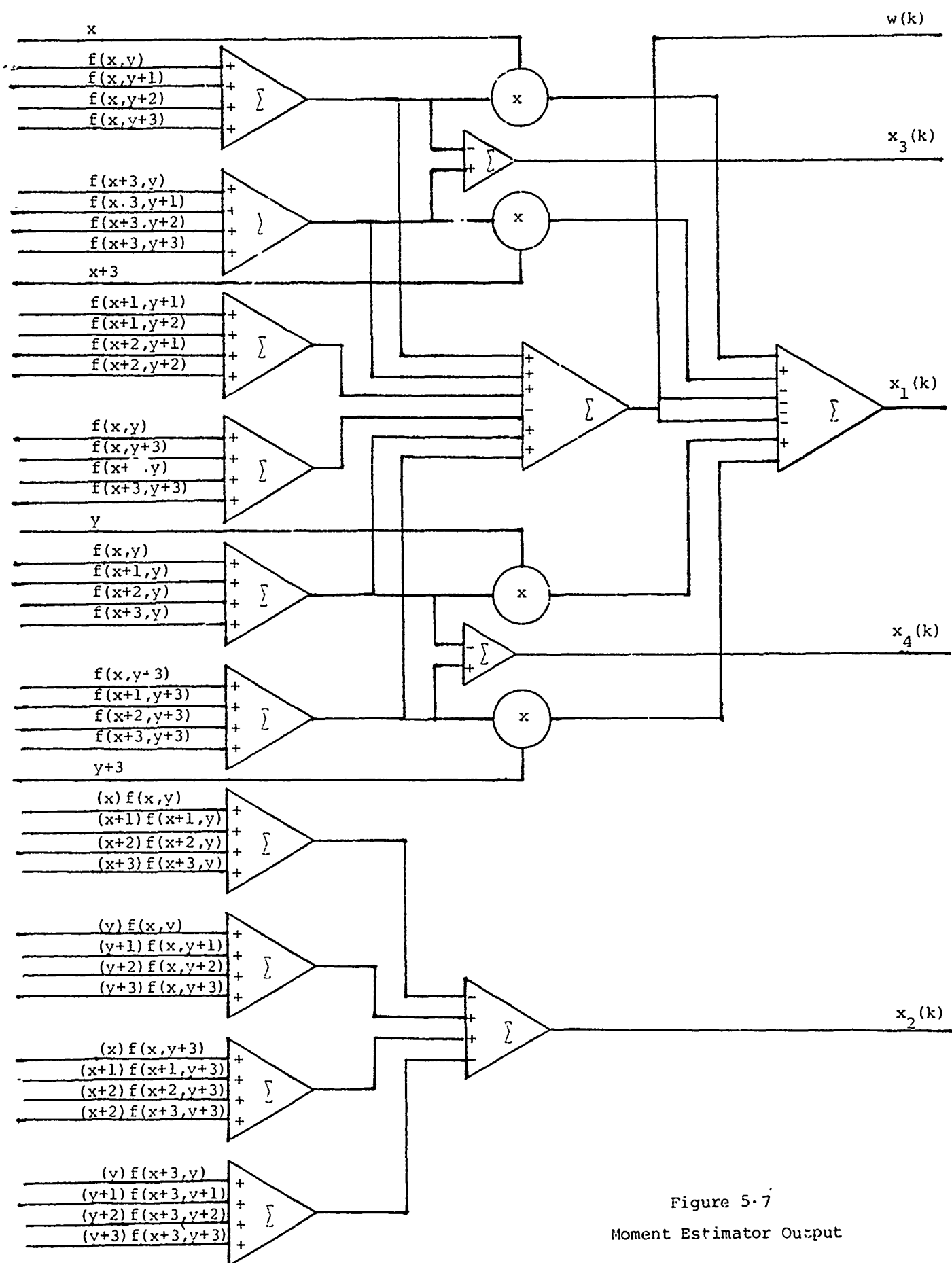


Figure 5-7
Moment Estimator Output

$$A(k) = [a_1(k,t) \ a_2(k,t) \ b_1(k,t) \ b_2(k,t)]^T \quad (5.3.4d)$$

$$x(k) = [x_1(k,t) \ x_2(k,t) \ x_3(k,t) \ x_4(k,t)]^T \quad (5.3.4e)$$

and

$$Z(k) = W(k,t) - W(k,t-T) \quad (5.3.4f)$$

Note that $A(0)$, $V_A(0)$, V_V and V_U must be initialized to some value before processing.

The complexity of this algorithm requires high speed processing. Therefore, special purpose digital hardware is used to implement this function. Due to the high speed requirements, as much of the processing as possible is performed in parallel. A diagram of the hardware implementation of this algorithm is shown in Figure 5-8.

5.3.5 Imager Controller

The imager controller consists of that hardware which controls the field of view for the image array. In this application, it is assumed that the image array is located in a small enclosure which can pan left and right or tilt up and down. Also a lens system allows zoom and focus control. The imager controller receives the parameter estimates from the parameter estimator, refines these estimates using the estimates from the previous frames, then generates the appropriate control signals to center the target within the field of view. The hardware involved in this implementation is beyond the scope of this investigation, and thus will not be presented in this section.

5.4 Summary

Several algorithms for performing image tracking have been presented in this section. A CCD based implementation with analog and digital support has been suggested for each algorithm, and thus each one appears to be feasible

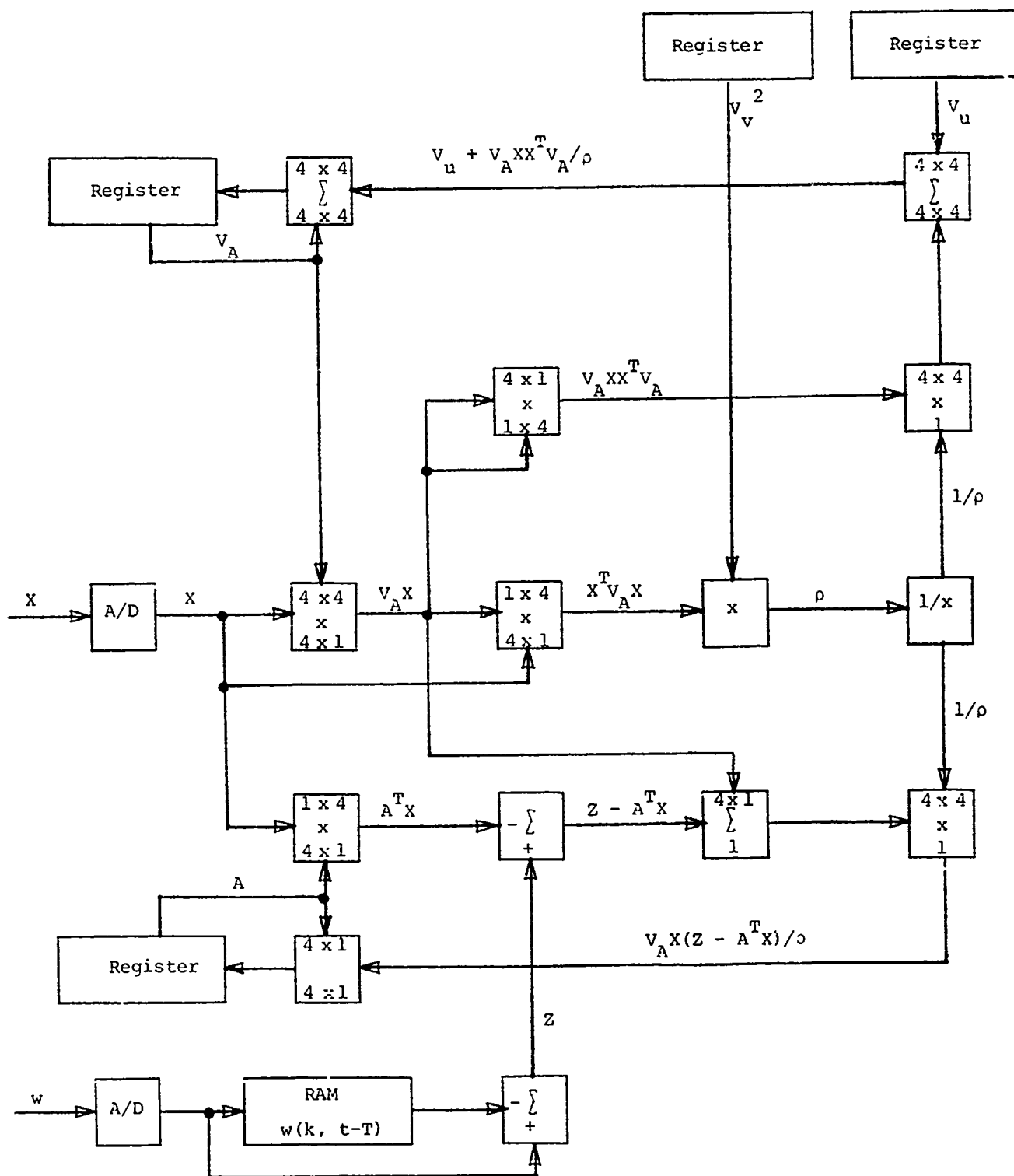


Figure 5-8

Digital Kalman parameter estimator

for use either as a substitute for the TSVIP algorithm or as a support algorithm for it. Further studies will be required to make this determination.

There are distinct advantages and disadvantages among the algorithms. The segmentation algorithms provide better lock capabilities than the gradient, moment, or coordinate based algorithms, since they provide absolute tracking measurements referenced to a fixed coordinate system as opposed to relative tracking measurement referenced to the previous frame. The transform based algorithms require more computations than the other algorithms. The sequential solution offered by the Kalman estimator or the Widrow-Hoff estimator has a more efficient hardware structure than a pseudo-inverse implementation, but a slow rate of convergence could easily offset this advantage. The segmentation and transform algorithms can tolerate a slower sampling rate even for fast targets, while the coordinate algorithms can tolerate a lower spatial sampling rate. All the algorithms can model rotation, dilation, and translation, but the segmentation algorithms are not affected by any movements. They are affected however by changes in texture or intensity. The gradient algorithms are very noise sensitive.

Thus there are many advantages and disadvantages that can now be observed among the possible algorithms. Table 5-1 roughly summarizes these, but the final decision must be based on simulation studies and hardware proposals. A program to simulate a moving target has been developed. This will provide a common data base to test the algorithms. The algorithms need to be developed as computer programs, and performance indices must be created to measure their performance with the common data base. A supplementary report on this task giving a summary of the performance data will be provided later, or the information will be available from a journal publication.

	GRADIENT	MOMENT	COORDINATE	TRANSFORMATION	SEGMENTATION
Computational Simplicity	4	4	2	2	3
CCD Implementability	5	5	4	3	5
Noise Immunity	1	3	4	3	4
Segmentation Immunity	4	4	1	4	2
Tracking Error Immunity	2	2	2	4	5
Spatial Sampling Immunity	2	3	4	3	5
Time Sampling Immunity	2	2	2	4	5

Legend:

- 1 = Poor
- 2 = Moderately Poor
- 3 = Fair
- 4 = Moderately Good
- 5 = Good

Table 5-1

Algorithm Comparison Chart

CHAPTER VI

RECOMMENDED FUTURE RESEARCH

Tracking and guidance using computer vision has much to offer in both military and civilian applications for a broad spectrum of activities. Systems that can look over the horizon and recognize specific patterns and then provide homing signals have many generic problems that are identical to those of systems which pick cabbages from a moving vehicle or perform tasks on the factory floor, so military technology will benefit from advanced automation and other computer vision research in progress both here and abroad. However, some research dedicated to tracking and guidance is required to be certain that problems specific to this area are solved in a timely manner.

Our results on CCD architectures for signal processing are encouraging. But, the practical and conceptual problems can be solved only through continued research. This technology provides one of the best possibilities for rapidly processing the very large quantities of data needed to realize high level, real time computer vision. The problems of size and reliability are of the utmost importance if systems are to be installed on small carriers and, here again, CCDs offer a possible solution. Separate research on CCD basic building blocks for signal processing is recommended if a unified research program is not continued.

The realization of practical computer vision tracking with a high degree of confidence is expected to require a hierarchy of algorithms working together to accommodate the wide variety of environmental conditions that can be encountered. The algorithms will include multiple solutions for measuring motion in three dimensions, segmenting, edging, recognition

and capability of accommodating multiple targets; and all of this in a noisy environment which partly, or occasionally completely, obscures the target. General algorithm work and testing will undoubtedly be a long term area of research, certainly in the next five to ten year time frame.

It is important that experimental work accompany the theoretical work. A computer vision tracking and guidance laboratory should be established to test and refine algorithms and to help delineate the problems that can be discovered only through hardware experimentation. The laboratory needs working systems and target simulation capability. We have reached the first performance level with our experimental system, but the components were originally built or acquired for manufacturing automation. Equipment designed for this specific problem is needed, including a dedicated digital computer. The establishment of such a laboratory is recommended because of the fundamental importance of small computer vision based tracking and guidance systems.

BIBLIOGRAPHY

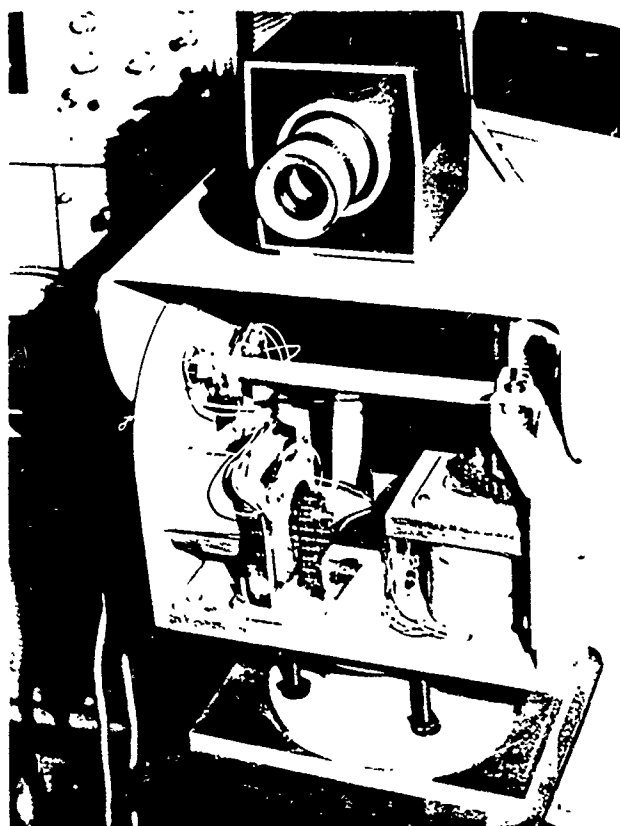
- [1] EG&G Reticon, Operation and Maintenance Manual MC5201 RS520 Camera/Controller System, 1976.
- [2] Hewlett-Packard Co., A Pocket Guide to the H.P. 2100 Computer, Sept. 1972.
- [3] Ahtam, R., "A computer Vision Control System," Master's Thesis, University of Virginia, Aug. 1974.
- [4] Goskez, A. K., "Picture Processing of Natural Scenes," Ph.D. Dissertation, University of Virginia, Aug. 1975.
- [5] Schalkoff, R. J., "Algorithms for a Real Time Automatic Video Tracking System," Ph.D. Dissertation, University of Virginia, May 1979.
- [6] Carroll, L. R., "A Novel Configuration for CCDs," Electronics Letters, Vol. 14, No. 7, March 1978, pp. 217, 219.
- [7] Mayer, D. J., Eversole, W. L. and Hewes, C. R., "A Programmable CCD Correlator for Pattern Classification," EG Engineering Journal, May-June 1980, pp. 13-18.
- [8] Tanaka, S., et. al., "An Integrated Real-Time Programmable Transversal Filter," EG&G Reticon Technical Report, 1980.
- [9] Young, I. A., "A High Performance All-Enhancement NMOS Operational Amplifier," IEEE JSSC, Vol. 14, N9. 6, Dec. 1979, pp. 1070, 1977.
- [10] Bosshart, P., "An Integrated Analog Correlator Using Charge-Coupled Devices," ISSCC Digest of Technical Papers, Philadelphia, Pa., Feb. 1976, pp. 198, 199.
- [11] EG&G Reticon, TAD-32 Tapped Analog Delay Line Technical Specifications brochure.
- [12] EG&G Reticon, R5501 32 Parallel In/ Serial Out Device Technical Specifications brochure.
- [13] Barbe, D. F., et. al., "Signal Processing with Charge-Coupled Devices," IEEE Transactions on Electron Devices, Vol. Ed-25, No. 2, Feb. 1978, pp. 108-125.
- [14] Barbe, D. F., et. al., "Signal Processing with Charge-Coupled Devices," IEEE Journal of Solid-State Circuits, Vol. SC-13, No. 1, Feb. 1978, pp. 34-51.
- [15] Steinberg, D. I., Computational Matrix Algebra, McGraw-Hill, N.Y., 1974.

- [16] Duda, R. O., and Hart, P. E., Pattern Classification and Scene Analysis John Wiley & Sons, Inc., N.Y., 1973.
- [17] Widrow, B., and Hoff, Jr., M., "Adaptive Switching Circuits," IRE WESCON Convention Record, Part 4, 1960, p. 96.
- [18] Morgan, D. R. and Craig, S. E., "Real-Time Adaptive Linear Prediction Using the Least Mean Square Gradient Algorithm," IEEE Transactions on Acoustics, Speech and Signal-Processing, Vol. ASSP-24, No. 6, Dec. 1976, pp. 494-507.
- [19] White, M. H, et. al., "CCD Adaptive Discrete Analog Signal Processing," IEEE Transactions on Communications, Vol. Com-27, No. 2, Feb. 1979, pp. 390-405.
- [20] Tou, J. T. and Gonzalez, R. C., Pattern Recognition Principles, Addison-Wesley Publishing Co., Reading, Mass., 1974.
- [21] Newman, Thomas G. and Demus, D. A., "Lie Theoretic Methods in Video Tracking," to be published (based on research contract N0014-76-C-1136), Texas Tech University, 1979.
- [22] Schalkoff, R. J. and McVey, E. S., "Algorithm Development for Real-Time Automatic Video Tracking Systems," COMPSAC Proceedings, November, 1979.
- [23] Decuchi, Koichiro and Morishita, Iwao, "Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques," IEEE Transactions on Computers, Vol. C-27, No. 8, Aug. 1978.

Appendix 2.1: Pin Connections for Video Interface Cable

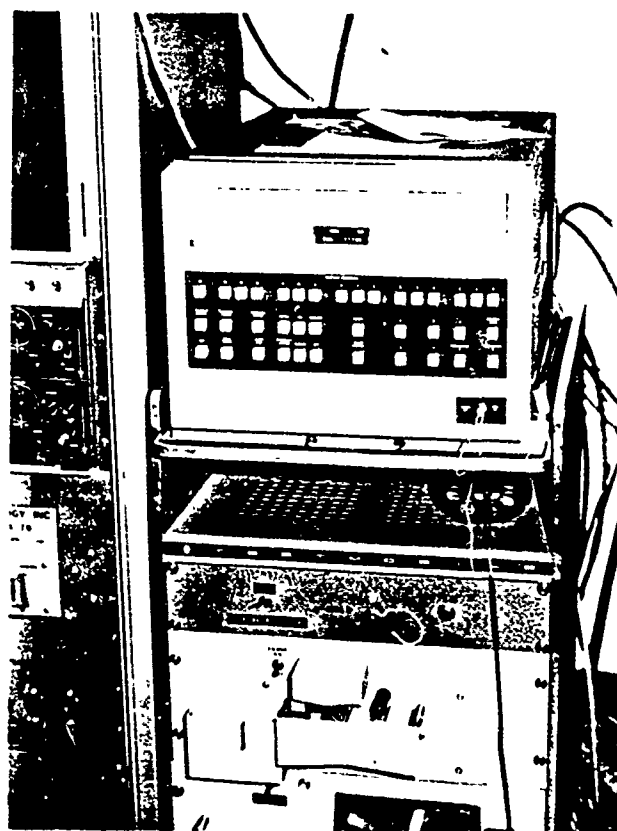
TO EXTERNAL DEVICE		
SIGNAL	PIN	DEVICE END CONTINENTAL CONNECTOR PIN
BIT 0	A	A
BIT 1	B	E
BIT 2	C	K
BIT 3	D	P
BIT 4	E	U
BIT 5	F	Y
BIT 6	H	c
BIT 7	J	h
BIT 8	K	n
BIT 9	L	t
BIT 10	M	x
BIT 11	N	BB
BIT 12	P	FF
BIT 13	R	C
BIT 14	S	H
BIT 15	T	M
DEVICE COMMAND	22,Z	k,r
GROUND	BB	DD

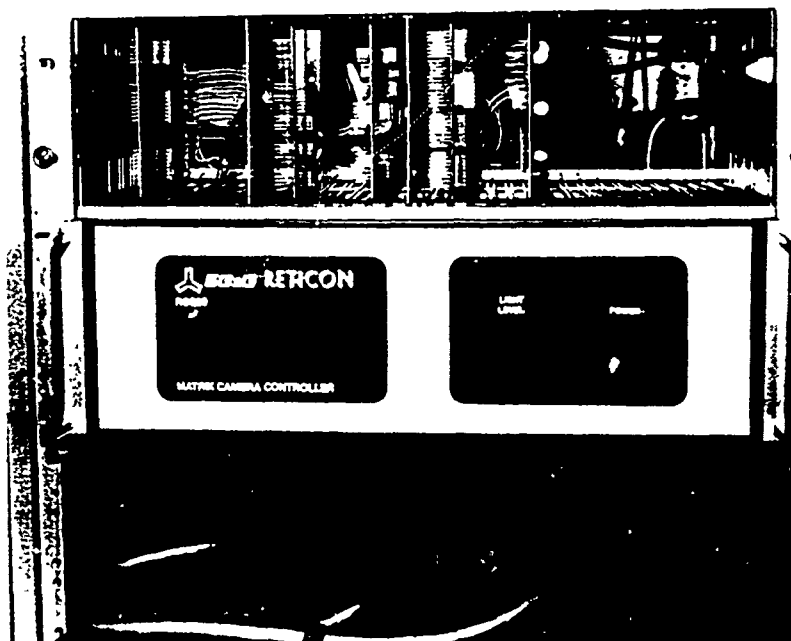
Appendix 2.2: Photographs of Tracking System



Photograph of camera
and pan tilt mount.

Photograph of HP2100
minicomputer.





Photograph of camera controller



Overall view of tracking system

FTM4 COMPILER: 4224177 (SEPT. 1974)

Best Available Copy

00057 C CURRENT CAMERA POSITION IS ASSUMED TO BE ORIGIN

```

0060 C
0061 C STORE ARRAY DUMP PARAMETERS
0062 DUMP(1)=IPOS1
0063 DUMP(2)=IPOS2
0064 DUMP(3)=JPOS1
0065 DUMP(4)=JPOS2
0066 C
0067 C HUMAN TARGET CONTROL
0068 IF (I'OVE.NE.0) GO TO 400
0069 CALL UTIL(1,PIC,DUMP,D1,D2,MEAN1,G,PC)
0070 WRITE (1,100)
0071 100 FORMAT ("ENTER 0 FOR SIMULATED AND 1 FOR REAL TARGET MOTION")
0072 READ (1,*) ITEST
0073 200 IF (ITEST.EQ.0) CALL PERT(PIC,I'MAX,J'MAX)
0074 IF (ITEST.EQ.1) CALL DRIVE
0075 CALL UTIL(2,PIC,DUMP,D1,D2,MEAN2,G,PC)
0076 SCALE=MEAN1/MEAN2
0077 CALL TRACK(K,PC,G,SCALE,D1,D2,A)
0078 WRITE (1,300) POS1,POS2,A(3,1),A(4,1)
0079 300 FORMAT (4F4.2)
0080 GO TO 200
0081 C
0082 C AUTOMATIC TARGET CONTROL
0083 400 IF (I'OVE.NE.1) GO TO 1200
0084 WRITE (1,500)
0085 500 FORMAT("ENTER DISPLACEMENT AND NUMBER OF AVERAGES")
0086 READ (1,*) JSTEP,JAVG
0087 WRITE (1,600)
0088 600 FORMAT ("ENTER 0 FOR HORIZONTAL OR 1 FOR VERTICAL DISPLACEMENT")
0089 READ (1,*) JDIS
0090 JVER=1
0091 JHOR=0
0092 IF (JDIS.NE.1) GO TO 700
0093 JVER=0
0094 JHOR=1
0095 700 DO 1000 J=1,JAVG
0096 CALL UTIL(1,PIC,DUMP,D1,D2,MEAN1,G,PC)
0097 DO 900 JJ=1,JSTEP
0098 CALL MOTOR(12,JVER*JSTEP*ISTEP1,JHOR*JSTEP*ISTEP2,3,3)
0099 CALL UTIL(2,PIC,DUMP,D1,D2,MEAN2,G,PC)
0100 SCALE=MEAN1/MEAN2
0101 CALL TRACK(K,PC,G,SCALE,D1,D2,A)
0102 WRITE (1,800) JJ,J,A(3,1),A(4,1),POS1,POS2
0103 800 FORMAT (2I6,2F4.2,2I6)
0104 PT(1,JJ)=(A(3,1)+(J-1)*PT(1,JJ))/J
0105 PT(2,JJ)=(A(4,1)+(J-1)*PT(2,JJ))/J
0106 900 CONTINUE
0107 CALL MOTOR(12,JVER*JSTEP*ISTEP1,JHOR*JSTEP*ISTEP2,3,3)
0108 POS1=0
0109 POS2=0
0110 CONTINUE
0111 WRITE (IODEV,1100) ((PT(I,J),I=1,2),J=1,JSTEP)
0112 1100 FORMAT (2F4.2)

```

PAGE 0003 CONT FTN4 COMPILER: MD24177 (SEPT. 1974)

```

0113 STOP
0114 C
0115 C CLOSED LOOP OPERATION
0116 1200 CALL UTIL(1,PIC,DUMP,D1,D2,MEAN1,G,PC)
0117 1300 CALL UTIL(2,PIC,DUMP,D1,D2,MEAN2,G,PC)
0118 SCALE=MEAN1/MEAN2
0119 CALL TRACK(K,PC,G,SCALE,D1,D2,A)
0120 CALL PAUT(A(3,1),A(4,1))

```

0122 1400 FORMAT (216)
0123 GO TO 1300
0124 C
0125 C
0126 END

A-7

** NO ERRORS** PROGRAM = 10913 COMMON = 00024

PAGE 0004 CONT FTN4 COMPILER: HP24177 (SEPT. 1974)

0127 ENDS

0055 COMMON JPOS1,JPOS2,IODEV,IDISP,IVIEW,IDOPT,IPOVE,PIAG,IP,JG
 0059 COMMON ISTEP1,ISTEP2,IDIR,POS1,POS2

A-9

```

0060 C
0061 C
0062 ISTART=1
0063 JSTART=1
0064 ISTOP=99
0065 JSTOP=99
0066 IDISP=0
0067 PIAG=2
0068 IP=0
0069 JG=0
0070 IVIEW=0
0071 IDOPT=1
0072 IPOVE=0
0073 IODEV=1
0074 C
0075 C
0076 WRITE (1,100)
0077 100 FORMAT("ENTER D FOR DEFAULT",//)
0078 C
0079 WRITE (1,200)
0080 200 FORMAT("ENTER ISTART,JSTART,ISTOP,JSTOP")
0081 READ (1,*) ISTART,JSTART,ISTOP,JSTOP
0082 C
0083 WRITE (1,300)
0084 300 FORMAT("ENTER DISPLAY OPTION: 0 NONE: 1 DISPLAY PICTURE: 2 DUMP")
0085 READ (1,*) IDISP
0086 IF (IDISP.NE.1) GO TO 400
0087 IF (IDISP.NE.1) GO TO 500
0088 WRITE (1,400)
0089 400 FORMAT("ENTER MAGNIFICATION AND COORDINATES OF SCENE")
0090 READ (1,*) PIAG,IP,JG
0091 500 IF (IDISP.NE.2) GO TO 700
0092 WRITE (1,600)
0093 READ (1,*) IPOS1,IPOS2,JPOS1,JPOS2
0094 600 FORMAT("ENTER LOCATION OF DATA TO BE VIEWED")
0095 READ (1,*) IPOS1,IPOS2,JPOS1,JPOS2
0096 C
0097 700 WRITE (1,800)
0098 800 FORMAT("ENTER DERR. DISPLAY OPTION: 0 NO DISPLAY: 1 DISPLAY")
0099 READ (1,*) IVIEW
0100 C
0101 WRITE (1,900)
0102 900 FORMAT("ENTER DERR. CALC. OPTION: 0 NORMAL: 1 FOR AVEPAGING")
0103 READ (1,*) IDOPT
0104 C
0105 WRITE (1,1000)
0106 1000 FORMAT("CONTROL OPTION: 0 OPERATOR: 1 AUTOMATIC: 2 CLOSED LOOP")
0107 READ (1,*) IPOVE
0108 C
0109 WRITE (1,1100)
0110 1100 FORMAT("ENTER OUTPUT DEVICE USED FOR CALCULATIONS")
0111 READ (1,*) IODEV
0112 C

```

PAGE 0003 INIT FTIA COMPILER: HP24177 (SEPT. 1974)

```

0113 WRITE (1,1200)
0114 1200 FORMAT("INITIAL STEP SIZE MOVEMENT OF CAMERA")
0115 READ (1,*) ISTEP1,ISTEP2
0116 C
0117 IMAX=ISTOP-ISTART+1
0118 JMAX=JSTOP-JSTART+1
0119 I=ISTART
0120 J=JSTART

```

0121 RETURN
0122 C
0123 C
0124 END

A-10

** NO ERRORS** PROGRAM = 00465 COMMON = 00024

PAGE 0004 INIT FTM4 COMPILER: HP24177 (SEPT. 1974)

0125 ENDS

PAGE 0002 UTIL FT14 COMPILER: 0824177 (SEPT. 1973)

035/ C REA) : REFAC' CATEDA APPAV

```

0060 C      CALL DLOAD(PIC,02)
0061 C      SECOND SCENE: STORE TARGET POINTS
0062 C      IF (IFLAG.EQ.2) CALL DLOAD(PIC,02)
0063 C
0064 C      FIRST SCENE: SEGMENT: STORE TARGET POINTS: CALCULATE DERRIVATIVES
0065 C      IF (IFLAG.NE.1) GO TO 100
0066 C      CALL RECT(PIC)
0067 C      CALL DLOAD(PIC,01)
0068 C      CALL DERIV(PIC,G,PC,X)
0069 C
0070 C      DUMP SECTOR OF DATA ARRAY
0071 100 IF (IDISP.NE.2) GO TO 700
0072 C      WRITE (1,200)
0073 C      WRITE (1,200) IPOS1,IPOS2,JPOS1,JPOS2
0074 200 FORMAT (4I6)
0075 C      DO 600 I=DUMP(1),DUMP(2)
0076 C      DO 400 J=DUMP(3),DUMP(4)
0077 C      WRITE (1,300) PIC(I,J)
0078 300 FORMAT (10I6)
0079 400 CONTINUE
0080 C      WRITE (1,500)
0081 500 FORMAT (//,"ROW ",I6)
0082 600 CONTINUE
0083 C
0084 C      CALCULATE RANGE ON PIC(I,J) IN ORDER TO DISPLAY IMAGE
0085 700 IF (IDISP.NE.1) GO TO 900
0086 C      PMIN=255
0087 C      PMAX=1
0088 C      DO 800 I=1,IMAX
0089 C      DO 300 J=1,JMAX
0090 C      IF (PIC(I,J).LT.PMIN) PMIN=PIC(I,J)
0091 C      IF (PIC(I,J).GT.PMAX) PMAX=PIC(I,J)
0092 C      TARGET BLANKING
0093 C      IF (IDISP.NE.3) GO TO 900
0094 C      PMIN=?
0095 C      IF (((I.LT.IPOS1).AND.(I.GT.IPOS2)).AND.((J.LT.JPOS1).AND.
0096 C      + (J.GT.JPOS2))) PIC(I,J)=0
0097 800 CONTINUE
0098 C      CALL DISP(PIC)
0099 C
0100 900 CONTINUE
0101 C
0102 C      RETURN
0103 C
0104 C
0105 C      END

```

** NO ERRORS ** PROGRAM = '031' COMMON = '0024'

PAGE 0003 UTIL FT14 COMPILER: HP24177 (SLP. 1974)

0106 ENDS

PAGE 0002 SORT FTA COMPILED: 4P24177 (SEPT. 1974)

0062	C	IF ROW POINTER EQUALS ZERO THEN RESET IT TO 100 AND	
0063	C	DECREMENT THE COLUMN POINTER	
0064		IF (JPOS.GT.0) GO TO 200	A-14
0065		JPOS=100	
0066		IPOS=IPOS-1	
0067	C		
0068	200	CONTINUE	
0069	C		
0070		RETURN	
0071	C		
0072	C		
0073		END	

** NO ERRORS** PROGRAM = 00124 COMMON = 00024

PAGE 0003 SORT FTR4 COMPILER: HP24177 (SEPT. 1974)

0074 ENDS


```

0057 C
0058 C   CREATE ROW SUM
0059 DO 100 I=1,IMAX
0060 DO 100 J=1,JMAX
0061     ISUM(I)=ISUM(I)+PIC(I,J)
0062 100 CONTINUE
0063 C
0064 C   CREATE COLUMN SUM
0065 DO 200 J=1,JMAX
0066 DO 200 I=1,IMAX
0067     JSUM(J)=JSUM(J)+PIC(I,J)
0068 200 CONTINUE
0069 C
0070 C   EXAMINE ROW AND COLUMN VECTORS FOR THEIR TWO GREATEST DIFFERENCES
0071 CALL SUM(ISUM,IMAX,IPOS1,IPOS2)
0072 CALL SUM(JSUM,JMAX,JPOS1,JPOS2)
0073 C
0074 C   SELECT COORDINATES OF 20 INNERMOST DATA POINTS
0075 CALL LOC
0076 C
0077 RETURN
0078 C
0079 C
0080 END

```

** NO ERRORS** PROGRAM = 00315 COMMON = 00324

PAGE 0000 RECT FTN4 COMPILER: UP24177 (SEPT. 1974)

0081 ENDS


```

0054 C      CALC. 1ST DIFFERENCE IS GREATER THAN SECOND LARGEST DIFFERENCE
0055 C      STORE NEW SECOND GREATEST DIFFERENCE AND POSITION
0056 1.00    IF (GRAD3.LE.GRAD2) GO TO 200
0057        GRAD2=GRAD3
0058        POS2=1
0059 C
0060 C
0061 2.00    CONTINUE
0062 C
0063 C      RETURN
0064 C
0065 C
0066 C
0067 C
0068 C
0069 C      END
** NO ERRORS**    PROGRAM = 00385    COMMON = 00000

```

A-18

PAGE 0003 SUB FT14 COMPILER: HP24177 (SEPT. 1974)

0070 ENDS

1062 IF (IPOS1.EQ.0) INUM=JNUM
 1063 SCALE=SQRT(IPROD/2.0)
 1064 C
 1065 C RETURN IF TOTAL NUMBER OF POINTS IS LESS THAN 20
 1066 IF (IPROD.LE.20) GO TO 3004
 1067 C
 1068 C ADJUST NUMBER OF POINTS IN EACH DIMENSION BY SCALE FACTOR
 1069 INUM=INT(INUM/SCALE)
 1070 JNUM=INT(JNUM/SCALE)
 1071 C
 1072 C CHECK SPECIAL CASE: INUM=0 OR JNUM=0
 1073 IF (INUM.NE.0) GO TO 3004
 1074 INUM=1
 1075 JNUM=20
 1076 3004 IF (JNUM.NE.0) GO TO 4004
 1077 JNUM=1
 1078 INUM=20
 1079 C
 1080 C COMPUTE NEW POSITIONS
 1081 4004 IPOS1=ITEMP+INUM/2
 1082 IPOS2=ITEMP-INUM/2
 1083 JPOS1=JTEMP+JNUM/2
 1084 JPOS2=JTEMP-JNUM/2
 1085 C
 1086 5004 RETURN
 1087 C
 1088 C
 1089 END

** NO EPILOGS** PROGRAM = 00157 COMMON = 00024

PAGE 0003 LOC FTN4 COMPILER: HP24177 (SEPT. 1974)

50 ENDS

PAGE 0001

FTM4 COMPILER: 4P24177 (SEPT. 1974)

```

0001      FTM4,L
0002      C
0003      C
0004      SUBROUTINE DLOAD(PIC,D)
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC      THE PURPOSE OF THIS ROUTINE IS TO CREATE A SCENE VECTOR THAT
0012      CC      CONTAINS THE POINTS WITHIN THE BOUNDARIES AS SPECIFIED BY
0013      CC      THE TARGET LOCATION IN THE INITIAL SCENE. THIS VECTOR WILL
0014      CC      LATER BE USED IN CALCULATING THE SCENE DIFFERENCE VECTOR.
0015      CC
0016      CC      SOURCE FILE:          DLOADS
0017      CC      OBJECT FILE:        DLOAD
0018      CC      CALLED BY:          MAIN
0019      CC
0020      CC      PIC                INTEGER DATA ARRAY
0021      CC      IPOS1 : IPOS2      TARGET COORDINATES IN X DIMENSION
0022      CC      JPOS1 : JPOS2      TARGET COORDINATES IN Y DIMENSION
0023      CC      D                  VECTOR CONTAINING POINTS INTERIOR
0024      CC                        TO INITIAL SCENE TARGET LOCATION
0025      CC      K                  TOTAL NUMBER OF POINTS
0026      CC      I , J              DO LOOP CONTROL VARIABLES
0027      CC
0028      CC
0029      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0030      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0031      C
0032      C
0033      INTEGER ISTART,JSTART,ISTOP,JSTOP,ICOUNT,IMAX,JMAX,IPOS1,IPOS2
0034      INTEGER JPOS1,JPOS2,IODEV,IODIS,IVIEF,IPOPT,IMOVE,PMAG,IO,JQ
0035      INTEGER ISTEP,ISTEP2,IOIR,POS1,POS2,I,J,K,PIC(100,100)
0036      REAL (20)
0037      COMMON ISTART,JSTART,ISTOP,JSTOP,ICOUNT,IMAX,JMAX,IPOS1,IPOS2
0038      COMMON JPOS1,JPOS2,IODEV,IODIS,IVIEF,IPOPT,IMOVE,PMAG,IO,JQ
0039      COMMON ISTEP,ISTEP2,IOIR,POS1,POS2
0040      C
0041      C
0042      C      STORE DATA POINTS IN D VECTOR
0043      I=0
0044      DO 1 I=IPOS2,IPOS1
0045      DO 1 J=JPOS2,JPOS1
0046      K=K+1
0047      D(K)=FLOAT(PIC(I,J))
0048      100 CONTINUE
0049      C
0050      RETURN
0051      C
0052      C
0053      C

```

** NO ERRORS**

PROGRAM = 4P250

COMMON = 4P24

04

00.6

A-22

0059 DO 1000 J=JA,JB,2
 0060 JL=(MOD((JB+JA-J+1),32)+96)*256
 0061 JH=((JB+JA-J+1)/32+32)*256
 0062 DO 1000 I=IA,IB,2
 0063 IL=(MOD((I-1),32)+64)*256
 0064 IH=((I-1)/32+32)*256
 0065 IN=(I-IA)/PMAG+1
 0066 JN=(J-JA)/PMAG+1
 0067 IP=(IPX(IN,JN)-PMIN)*16/(PMAX-PMIN+1)
 0068 IF(IP.GT.15) IP=15
 0069 IF(IP.LE.0) GOTO 1000
 0070 WRITE (15,1) GRAY(IP),JH,JL,IH,IL,24329
 0071 1000 CONTINUE
 0072 C
 0073 WRITE(15,1) 7936
 0074 C
 0075 RETURN
 0076 C
 0077 C
 0078 END

A-24

** NO ERRORS** PROGRAM = 00349 COMMON = 00024

PAGE 0003 DISP FTM4 COMPILER: HP24177 (SEPT. 1974)

0079 ENDS


```

0001      I1=I+1
0002      I2=J
0003      I3=I
0004      I4=J
0005      I5=J+1
0006      I6=J
0007      I7=I
0008      GO TO 200
0009      C
0010      C      DERIVATIVE USING THE I+1 AND I-1 POINTS
0011      I1=I+1
0012      I2=J
0013      I3=I-1
0014      I4=J
0015      I5=J+1
0016      I6=J-1
0017      I7=2
0018      C
0019      C      LOAD THE ARRAYS
0020      K=K+1
0021      G(K,1)=FLOAT(PIC(I1,I2)-PIC(I3,I2))/I7
0022      G(K,2)=FLOAT(PIC(I4,I5)-PIC(I4,I6))/I7
0023      PC(K,1)=(I-IORG)*G(K,1)+(J-JORG)*G(K,2)
0024      PC(K,2)=(I-IORG)*G(K,2)-(J-JORG)*G(K,1)
0025      C
0026      300 CONTINUE
0027      C
0028      C
0029      C      IF (I+1 .EQ. 1) GO TO 700
0030      C
0031      C      OUTPUT DERIVATIVE CALCULATIONS
0032      WRITE (1,*)
0033      WRITE (1,*) ((G(I,J),J=1,2),I=1,K)
0034      WRITE (1,*)
0035      WRITE (1,*) ((PC(I,J),J=1,2),I=1,K)
0036      400 FORMAT (///,"G ARRAY",/)
0037      500 FORMAT (2,1,2)
0038      600 FORMAT (///,"PC ARRAY",/)
0039      C
0040      700 STOP
0041      C
0042      C
0043      C

```

** 10 PRINTSAL PROGRAM = 002417 COMMON = 00024
 PAGE 103 DEBY FTN COMPILER: 0024177 (SEPT. 1974)

1 4 0010

0058 RETURN
0059 C
0060 C
0061 END

A-28

** NO ERRORS** PROGRAM = 00015 COMMON = 00000

PAGE 0003 TRACK FT14 COMPILER: HP24177 (SEPT. 1974)

0002 ETL5

0060 C
0061

END

A-30

** NO ERRORS** PROGRAM = 00078 COMMON = 00000

PAGE 0003 PSEUD FTN4 COMPILER: UP24177 (SEPT. 1974)

0062 ENDS

PAGE 0001

FTH4 COMPILER: HP24177 (SEPT. 1974)

```

0001      FTH4,L
0002      C
0003      C
0004      SUBROUTINE INV(A,IA,N,IFAIL)
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC      THE PURPOSE OF THIS ROUTINE IS TO FIND THE INVERSE OF A MATRIX
0012      CC      BY GAUSS-JORDAN ELIMINATION. APPLYING TO THE ROWS OF THE
0013      CC      IDENTITY MATRIX I THE SAME ELEMENTARY ROW TRANSFORMATIONS AS
0014      CC      THOSE BY WHICH A IS REDUCED TO THE CANONICAL FORM I, THE
0015      CC      INVERSE OF A IS OBTAINED. PARTIAL PIVOTING FOR THE LARGEST
0016      CC      DIAGONAL ELEMENT IS ALSO DONE.
0017      CC
0018      CC      SOURCE FILE:      INVS
0019      CC      OBJECT FILE:     INVR
0020      CC      CALLED BY:      PSEUD
0021      CC
0022      CC      A      AN N X N INPUT ARRAY WHOSE INVERSE IS
0023      CC      DESIRED. NOTE A IS DESTROYED BY ITS
0024      CC      INVERSE.
0025      CC      IA     COLUMN LENGTH OF A
0026      CC      N      THE ORDER OF THE SYSTEM
0027      CC      IROW   TEMPORARY STORAGE ARRAY OF N WORDS
0028      CC      TEMP    TEMPORARY STORAGE ARRAY OF N WORDS
0029      CC
0030      CC
0031      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0032      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0033      C
0034      C
0035      DIMENSION A(IA,IA),IROW(100),TEMP(100),PI(5)
0036      C
0037      C
0038      IFAIL=0
0039      K=N-1
0040      DO 14 L=1,K
0041      L'=K-L+1
0042      A'AX=ABS(A(L,L))
0043      ISAVE=L
0044      L'Z=L+1
0045      DO 19 J=L'Z,N
0046      IF(ABS(A(J,L)).LE. A'AX) GO TO 18
0047      A'AX=ABS(A(J,L))
0048      ISAVE=J
0049      IF CONTINUE
0050      IF (A'AX.EQ.0) GO TO 100
0051      IROW(L)=ISAVE
0052      DO 20 M=1,L'
0053      X=A(L,M)
0054      A(L,M)=A(ISAVE,M)
0055      A(ISAVE,M)=X
0056      DO 14 L=1,L'

```

PAGE 0002 INV

FTH4 COMPILER: HP24177 (SEPT. 1974)

```

0057      LS=I+1
0058      A(LS,L)=-A(LS,L)/A(L,L)
0059      DO 14 II=1,K
0060      LT=II+1
0061      IF (LT.GT.L) GO TO 14
0062      LT=LT-1
0063      14  A(LS,LT)=A(LS,LT)+A(LS,L)*A(L,LT)
0064      IF (A(L,II).EQ.0) GO TO 140
0065      DO 64 I=1,K
0066      64  A(I,I)=A(I,I)/A(I,II)
0067      A(I,N)=1./A(I,II)
0068      DO 32 L=1,N
0069      LM=K-L+1
0070      LLL=L+1
0071      DO 24 J=1,LLL
0072      LT=N-J+1
0073      TEMP(J)=A(LM,LT)
0074      A(LM,LT)=0.0
0075      IF (J.EQ.LLL) A(LM,LT)=1.0
0076      24  CONTINUE
0077      DO 30 M=1,L
0078      M2=M+1
0079      DO 30 N1=1,N
0080      LZ=N-N1+1
0081      30  A(L,M,LZ)=A(LM,LZ)-TEMP(M)*A(M2,LZ)
0082      DO 32 III=1,N
0083      32  A(L,III)=A(LM,III)/TEMP(L+1)
0084      DO 50 I=1,K
0085      LM=K-I+1
0086      IF (IRON(LM).EQ.LM) GO TO 50
0087      NI=IRON(LM)
0088      DO 52 J=1,N
0089      X=A(J,LM)
0090      A(J,L)=A(J,NI)
0091      52  A(J,NI)=X
0092      50  CONTINUE
0093      RETURN
0094      100 IFAIL=1
0095      RETURN
0096      END

```

** NO ERRORS** PROGRAM = 00906 COMMON = 00000

PAGE 0003 INV FTN4 COMPILER: 0024177 (SEPT. 1974)

0097 END6


```

0001      .FTM4,L
0002      C
0003      C
0004      SUBROUTINE DRIVE
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC    THE PURPOSE OF THIS ROUTINE IS TO ALLOW THE OPERATOR TO MOVE OR
0012      CC    POSITION THE CAMERA SENSOR SOME SPECIFIED AMOUNT.
0013      CC
0014      CC    SOURCE FILE:          DRIVS
0015      CC    OBJECT FILE:         DRVR
0016      CC    CALLED BY:           CONT
0017      CC
0018      CC    THIS PROGRAM CALLS THE FOLLOWING ROUTINES:
0019      CC        MOTOR          ASSEMBLY LANGUAGE DRIVER FOR PANT TILT
0020      CC                        MOUNT
0021      CC
0022      CC    ISTEP1             NUMBER OF MOTOR STEPS IN Y DIMENSION
0023      CC                        1 STEP = 0.73 DEGREES
0024      CC    ISTEP2             NUMBER OF MOTOR STEPS IN X DIMENSION
0025      CC                        1 STEP = 0.46 DEGREES
0026      CC    IDIR               DIRECTION OF ROTATION
0027      CC                        0     CW TILT: CCW PAN
0028      CC                        1     CW TILT: CCW PAN
0029      CC                        2     CCW TILT: CW PAN
0030      CC                        3     CW TILT: CW PAN
0031      CC    IPR              MOTOR SPEED: 1 LOW; 2 MEDIUM; 3 HIGH
0032      CC    POS1 & POS2       CURRENT CAMERA POSITION
0033      CC
0034      CC
0035      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0036      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0037      C
0038      C
0039      INTEGER ISTART,JSTART,ISTOP,JSTOP,ICOUNT,IMAX,JMAX,IPOS1,IPOS2
0040      INTEGER JPOS1,JPOS2,IODEV,IOISP,IVIEW,IOOPT,IMOVE,PWAG,IA,JA
0041      INTEGER ISTEP1,ISTEP2,IDIR,POS1,POS2
0042      COMMON ISTART,JSTART,ISTOP,JSTOP,ICOUNT,IMAX,JMAX,IPOS1,IPOS2
0043      COMMON JPOS1,JPOS2,IODEV,IOISP,IVIEW,IOOPT,IMOVE,PWAG,IA,JA
0044      COMMON ISTEP1,ISTEP2,IDIR,POS1,POS2
0045      C
0046      C
0047      C    INITIALIZE PAN TILT PARAMETERS
0048      WRITE(1,10)
0049      10   FORMAT("ENTER #STEPS FOR PAN AND TILT AND DIRECTION")
0050      READ(1,*) ISTEP1,ISTEP2,IDIR
0051      IPR=3
0052      C
0053      C    CALL DRIVER TO MOVE PAN TILT
0054      CALL MOTOR(12,ISTEP1,ISTEP2>IDIR,IPR)
0055      C
0056      C    CALCULATE NEW CAMERA POSITION

```

PAGE 0002 DRIVE FTN4 COMPILER: 0524177 (SEPT. 1974)

```

0057      IF ((IDIR.EQ.0).OR.(IDIR.EQ.2)) ISTEP1=-ISTEP1
0058      IF ((IDIR.EQ.1).OR.(IDIR.EQ.3)) ISTEP2=-ISTEP2

```

0060 POS2=POS2+1STEP2
0061 C
0062 RETURN
0063 C
0064 C
0065 END

A-36

** NO ERRORS** PROGRAM = 03102 COMMON = 00024

PAGE 0003 DRIVE FTN4 COMPILE: HP24177 (SEPT. 1974)

0066 ENDS

PAGE 0001

0001

ASMB,L,R,T

CAS R.00000016

.ENTR X 0000001

PAR R 0000000

ORBIT R 0000001

IODEV R 0000002

MASK R 0000003

XSTRT R 0000004

YSTRT R 0000005

YSTOP R 0000006

OUT 0000032

** NO ERRORS*

PAGE 0002 #01

0001

ASMB,L,P,T

0002 000000

NAM CAS.7

0003

ENT CAS

0004

EXT .ENTR

0005

BSS 6

0006

NOP

0007

JSB .ENTR

0008

DEF PAR

0009

CLF 0

0010

LDA IODEV

0011

OTA 6

0012

CLC 2

0013

LDA PAR+5

0014

IOR ORBIT

0015

OTA 2

0016

STC 2

0017

LDA PAR+4,I

0018

CMA

0019

INA

0020

OTA 2

0021

LDA PAR,I

0022

AND MASK

0023

IOR XSTRT

0024

OTA OUT

0025

STC OUT

0026

CLC OUT

0027

LDA PAR+1,I

0028

AND MASK

0029

IOR YSTRT

0030

OTA OUT

0031

STC OUT

0032

CLC OUT

0033

LDA PAR+2,I

0034

OTA OUT

0035

STC OUT

0036

CLC OUT

0037

LDA PAR+3,I

0038

AND MASK

0039

IOR YSTOP

0040

OTA OUT

0041

STC OUT

Best Available Copy

0044	00054	122376	SBS	65
0045	00055	120754R	JMP	*-1
0046	00056	140776	CLC	2
0047	00057	102100	STP	2
0048	00060	120000R	JMP	045.1
0049	00061	120000R	OPRIT	OCT 1000000
0050	00062	030032	ICDEFV	OCT 32
0051	00063	030177	MASK	OCT 177
0052	00064	000400	XSTRT	OCT 400
0053	00065	000200	YSTRT	OCT 200
0054	00066	001600	YSTOP	OCT 1600
0055	00032		OUT	FOU 328
0056			END	

A-38

PAGE 0003 #01

** NO ERRORS*

PAGE 0001

ASMR.L.R.T

0001
 MOTOR R 000005
 .ENTR X 000001
 PAR R 000000
 PUL1 R 000043
 MOTO1 R 000054
 ZERO1 R 000072
 MOT2 R 000136
 PUL2 R 000114
 MOTO2 R 000124
 ZERO2 R 000142
 FIN R 000156
 P1 R 000161
 P2 R 000162
 P3 R 000163
 P4 R 000164
 M011 R 000165
 M012 R 000166
 M021 R 000167
 M022 R 000170
 ISTE1 R 000171
 ISTE2 R 000172
 IDIR R 000173
 IPER R 000174
 PER R 000175
 PER1 R 000176
 PER2 R 000177
 PER3 R 000207
 PER12 R 000201
 PER22 R 000202
 PER32 R 000203
 STE1 R 000204
 PE2 R 000205
 ZE2 R 000206
 STE2 R 000207
 PE1 R 000210
 ZE1 R 000211
 LOOP1 R 000212
 LOOP2 R 000213
 A/D 000016
 ** 10 ERRORS **

Best Available Copy

PAGE 0002

ASMR.L.R.T

0001			
0002	000000		HAD MOTOR, 7
0003			EXT MOTOR
0004			EXT .ENTR
0005	000000	PAR	BSS 5
0006	000000	MOTOR	HOP
0007	000000		JSB .ENTR
0008	000000		OFF PAR
0009	000000		OFF
0010	000000		LOA PAR+1.1
0011	000000		STA ISTE1
0012	000000		ADA PI
0013	000000		CMA

0016	00017	020134	LDA	200+3.1	
0017	00020	072174R	STA	1054	
0018	00021	052151R	CPA	21	
0019	00022	020133R	JMP	*+5	
0020	00023	052152R	CPA	22	
0021	00024	020144R	JMP	*+12	
0022	00025	052210R	LDA	200R3	
0023	00026	072212R	STA	10001	
0024	00027	052213R	LDA	20032	
0025	00030	072213R	STA	10002	
0026	00031	020143R	JMP	2011	
0027	00032	052170R	LDA	2021	
0028	00033	072212R	STA	10001	
0029	00034	052211R	LDA	20012	
0030	00035	072213R	STA	10002	
0031	00036	020143R	JMP	2011	
0032	00037	052177R	LDA	2002	
0033	00040	072212R	STA	10001	
0034	00041	052212R	LDA	20022	
0035	00042	072213R	STA	10002	
0036	00043	162113R	FUL1	LDA	200+3.1
0037	00044	072173R	STA	1011	
0038	00045	012151R	ADA	21	
0039	00046	052151R	CPA	21	
0040	00047	020051R	JMP	*+2	
0041	00050	020053R	JMP	*+3	
0042	00051	052155R	LDA	2011	
0043	00052	020054R	JMP	20101	
0044	00053	052156R	LDA	2012	
0045	00054	030214R	NOT01	ISZ	201
0046	00055	020057R	JMP	*+2	
0047	00056	020116R	JMP	2012	
0048	00057	132616	OTA	AZD	
0049	00060	062212R	LDA	10001	
0050	00061	072211R	STA	201	
0051	00062	030211R	ISZ	201	
0052	00063	020055R	JMP	*+2	
0053	00064	020072R	JMP	ZERO1	
0054	00065	062213R	LDA	10002	
0055	00066	072215R	STA	202	
0056	00067	036215R	ISZ	202	

PAGE 0003 001

0057	00070	020057R	JMP	*-1	
0058	00071	020062R	JMP	*-7	
0059	00072	032400	ZERO1	CLA	
0060	00073	132616	OTA	AZD	
0061	00074	062212R	LDA	10001	
0062	00075	072211R	STA	201	
0063	00076	030211R	ISZ	201	
0064	00077	020111R	JMP	*+2	
0065	00100	020043R	JMP	2011	
0066	00101	062213R	LDA	10002	
0067	00102	072216R	STA	202	
0068	00103	030216R	ISZ	202	
0069	00104	020113R	JMP	*-1	
0070	00105	020076R	JMP	*-7	
0071	00106	162012R	NOT2	LDA	200+2.1
0072	00107	072172R	STA	1052	
0073	00110	042151R	ADA	21	
0074	00111	033000	CPA		
0075	00112	052004	INA		
0076	00113	072217R	STA	202	
0077	00114	020173R	2012	LDA	1011

0080	00117	020121R	JMP **2
0081	00120	020123R	JMP **3
0082	00121	020124R	LDA M021
0083	00122	020124R	JMP MOTO2
0084	00123	020127R	LDA M022
0085	00124	030227R	MOTO2 ISZ STE2
0086	00125	020127R	JMP **2
0087	00126	020128R	JMP FIN
0088	00127	102016	OTA A/D
0089	00131	062212R	LDA LOOP1
0090	00131	072211R	STA PE1
0091	00132	030210R	ISZ PE1
0092	00133	020135R	JMP **2
0093	00134	020142R	JMP ZER02
0094	00135	062213R	LDA LOOP2
0095	00136	072205R	STA PE2
0096	00137	030205R	ISZ PE2
0097	00140	020137R	JMP *-1
0098	00141	020132R	JMP *-7
0099	00142	002400	ZER02 CLA
0100	00143	102016	OTA A/D
0101	00144	062213R	LDA LOOP1
0102	00145	072211R	STA ZE1
0103	00146	030211R	ISZ ZE1
0104	00147	020131R	JMP **2
0105	00150	020114R	JMP PUL2
0106	00151	062213R	LDA LOOP2
0107	00152	072206R	STA ZE2
0108	00153	030206R	ISZ ZE2
0109	00154	020133R	JMP *-1
0110	00155	020146R	JMP *-7
0111	00156	062104R	FIN LDA P4
0112	00157	102103	STF 1

PAGE 0004 001

0113	00160	120005R	JMP MOTOR.1
0114	00161	000001	P1 OCT 1
0115	00162	000002	P2 OCT 2
0116	00163	000003	P3 OCT 3
0117	00164	000004	P4 OCT 4
0118	00165	000005	M011 OCT 10000
0119	00166	020000	M012 OCT 20000
0120	00167	040000	M021 OCT 40000
0121	00170	100000	M022 OCT 100000
0122	00171	000000	ISZ PE1 NOP
0123	00172	000000	ISZ PE2 NOP
0124	00173	000000	ISZ ZE1 NOP
0125	00174	000000	ISZ ZE2 NOP
0126	00175	175272	PER DEC -1350
0127	00176	175272	PER1 DEC -1350
0128	00177	177272	PER2 DEC -350
0129	00200	177545	PER3 DEC -137
0130	00201	177770	PER12 DEC -2
0131	00202	177777	PER22 DEC -1
0132	00203	177777	PER32 DEC -1
0133	00204	000000	STE1 NOP
0134	00205	000000	PE2 NOP
0135	00206	000000	ZE2 NOP
0136	00207	000000	STE2 NOP
0137	00210	000000	PE1 NOP
0138	00211	000000	ZE1 NOP
0139	00212	000000	LOOP1 NOP
0140	00213	000000	LOOP2 NOP

Best Available Copy

** NO ERRORS *

A-42

PAGE 0001

FTN4 COMPILER: 4P24177 (SEPT. 1974)

```

0001      FTM4,L
0002      C
0003      C
0004      SUBROUTINE CLR(A,M,N)
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC      THE PURPOSE OF THIS ROUTINE IS TO INITIALIZE A REAL ARRAY
0012      CC      TO ZERO.
0013      CC
0014      CC      SOURCE FILE:          CLPS
0015      CC      OBJECT FILE:         CLR2
0016      CC      CALLED BY:           KPY
0017      CC
0018      CC      A                      REAL DATA ARRAY
0019      CC      M & N                DIMENSIONS OF THE ARRAY
0020      CC      I & J                DO LOOP CONTROL VARIABLES
0021      CC
0022      CC
0023      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0024      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025      C
0026      C
0027      INTEGER I,J,M,N
0028      REAL A(M,N)
0029      C
0030      C
0031      DO 100 I=1,M
0032      DO 100 J=1,N
0033      A(I,J)=0
0034      100 CONTINUE
0035      C
0036      RETURN
0037      C
0038      C
0039      END

```

```

** NO ERROR**      PROGRAM = 00054      COMMON = 00000
PAGE 0002 CLR      FTM4 COMPILER: 4P24177 (SEPT. 1974)

```

0040 ENDS

PAGE 0001

FTN4 COMPILER: 4P24177 (SEPT. 1974)

```

0001      FTN4,L
0002      C
0003      C
0004      SUBROUTINE MPY(A,B,C,M,N,P)
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC      THE PURPOSE OF THIS ROUTINE IS TO PERFORM MATRIX MULTIPLICATION
0012      CC      ON THE FIRST TWO ARRAYS AND STORE THE RESULT IN THE THIRD ARRAY
0013      CC      [MAT C = MAT A * MAT B].
0014      CC
0015      CC      SOURCE FILE:      MPYS
0016      CC      OBJECT FILE:     MPYR
0017      CC      CALLED BY:      MAIN
0018      CC
0019      CC      A      REAL ARRAY: 1ST OPERAND: DIMENSION M X N
0020      CC      B      REAL ARRAY: 2ND OPERAND: DIMENSION N X P
0021      CC      C      REAL ARRAY: RESULT: DIMENSION M X P
0022      CC
0023      CC      M,N,P      DIMENSION OF ARRAYS SPECIFIED ABOVE
0024      CC      I,J,K      DO LOOP CONTROL VARIABLES
0025      CC
0026      CC
0027      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0028      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0029      C
0030      C
0031      INTEGER I,J,K,M,N,P
0032      REAL A(M,N),B(N,P),C(M,P)
0033      C
0034      C
0035      C      INITIALIZE RESULT
0036      C      CALL CLR(C,M,P)
0037      C
0038      C      PERFORM MATRIX MULTIPLICATION
0039      C      DO 100 P=1,P
0040      C      DO 120 I=1,P
0041      C      DO 130 J=1,N
0042      C      C(I,I)=C(K,I)+A(K,J)*B(J,I)
0043      C      CONTINUE
0044      C
0045      C      RETURN
0046      C
0047      C
0048      END

```

** NO ERRORS** PROGRAM = 00102 COMMON = 00100

PAGE 0002

MPY

FTN4 COMPILER: 4P24177 (SEPT. 1974)

0049

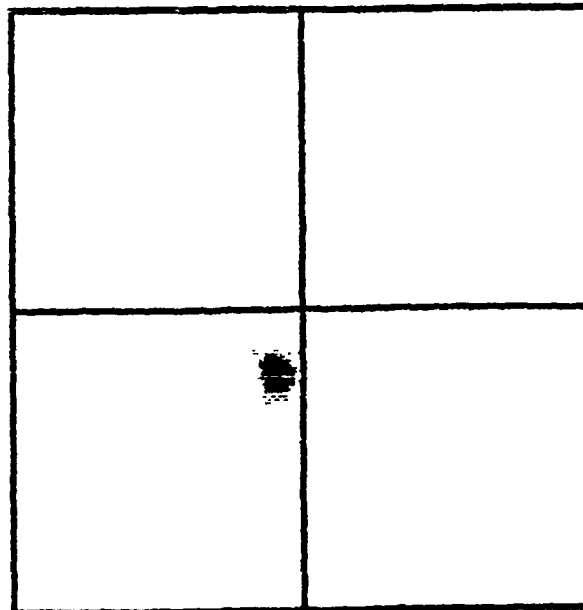
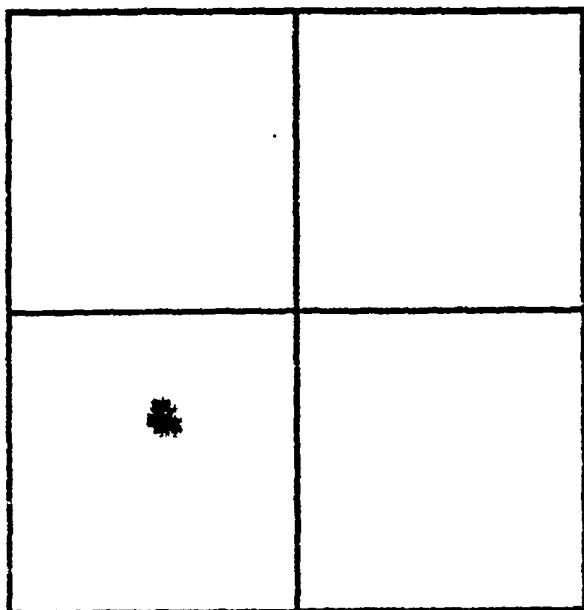
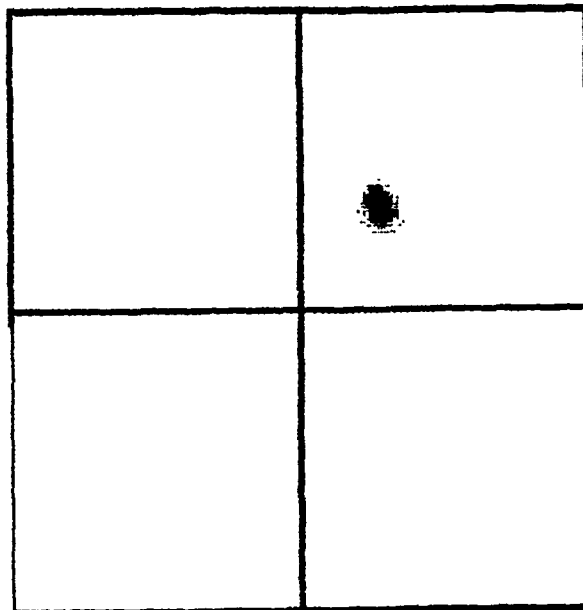
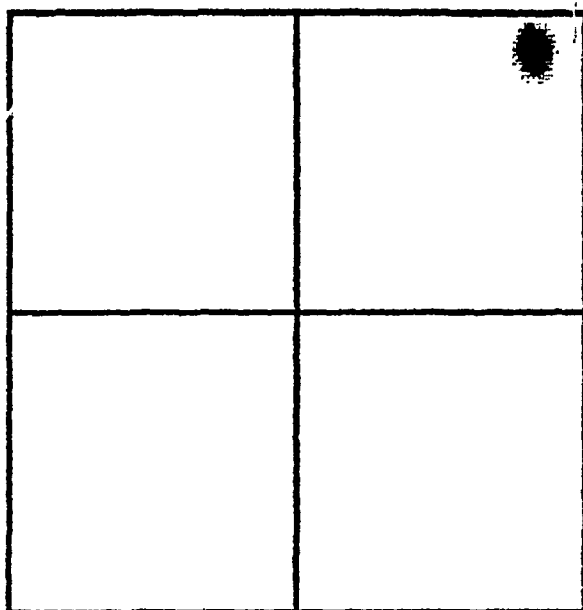
ENDS

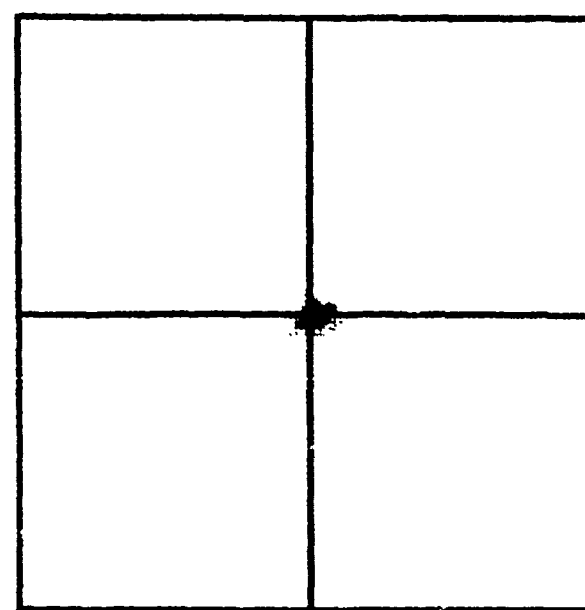
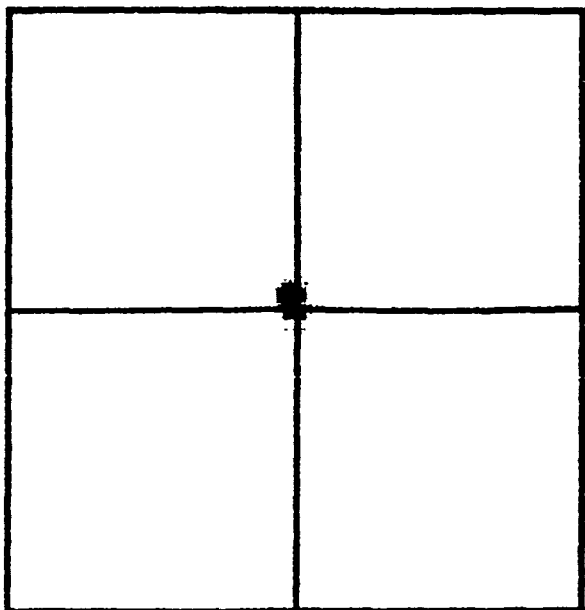
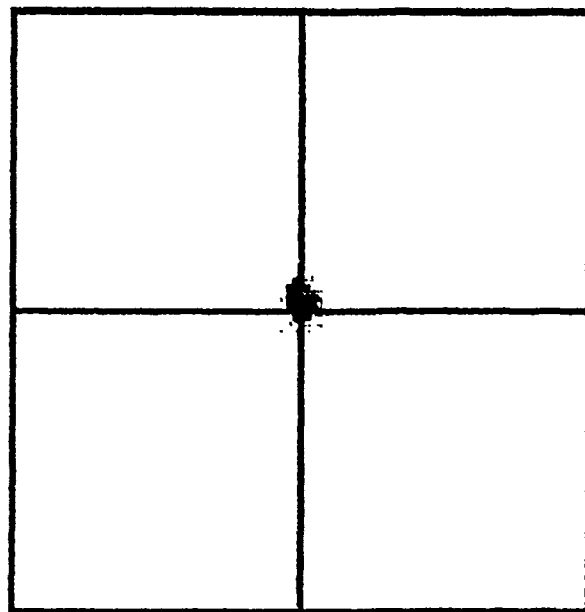
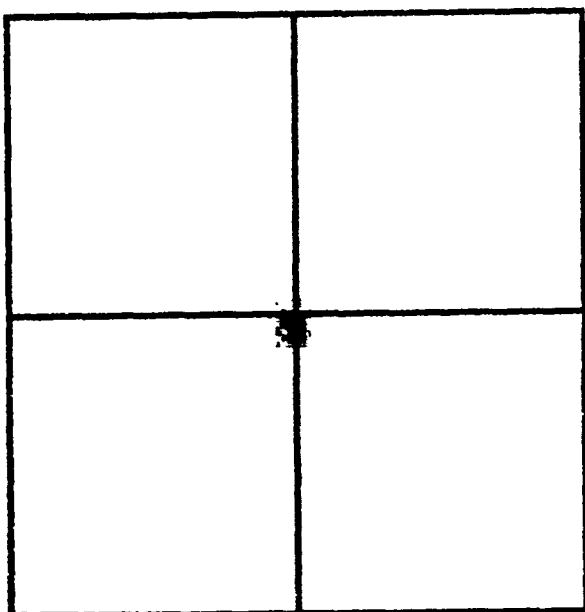

```
0001      FTH4.L
0002      C
0003      C
0004      SUBROUTINE SUB(A,B,C,"N)
0005      C
0006      C
0007      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0009      CC
0010      CC
0011      CC      THE PURPOSE OF THE ROUTINE IS TO GENERATE MATRIX SUBTRACTION
0012      CC      OF THE FORM C = A - B
0013      CC
0014      CC      SOURCE FILE:          SUBS
0015      CC      OBJECT FILE:         SUBR
0016      CC      CALLED BY:           TRACK
0017      CC
0018      CC      A, B, & C          REAL ARRAYS WITH SUBTRACTION DEFINED AS
0019      CC                          FOLLOWS C = A - B
0020      CC      M & N              DIMENSIONS OF THE TWO ARRAYS
0021      CC      I & J              DO LOOP CONTROL VARIABLES
0022      CC
0023      CC
0024      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0025      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0026      C
0027      C
0028      INTEGER I,J,K,L,M
0029      REAL A(I,N),B(K,L),C(M,N)
0030      C
0031      DO 100 I=1,M
0032      DO 100 J=1,N
0033      100 C(I,J)=A(I,J)-B(I,J)
0034      C
0035      RETURN
0036      C
0037      C
0038      END
```

** NO ERRORS** PROGRAM = 00075 COMMON = 00000

PAGE 0002 SUB FTH4 COMPILER: HP24177 (SEPT. 1974)

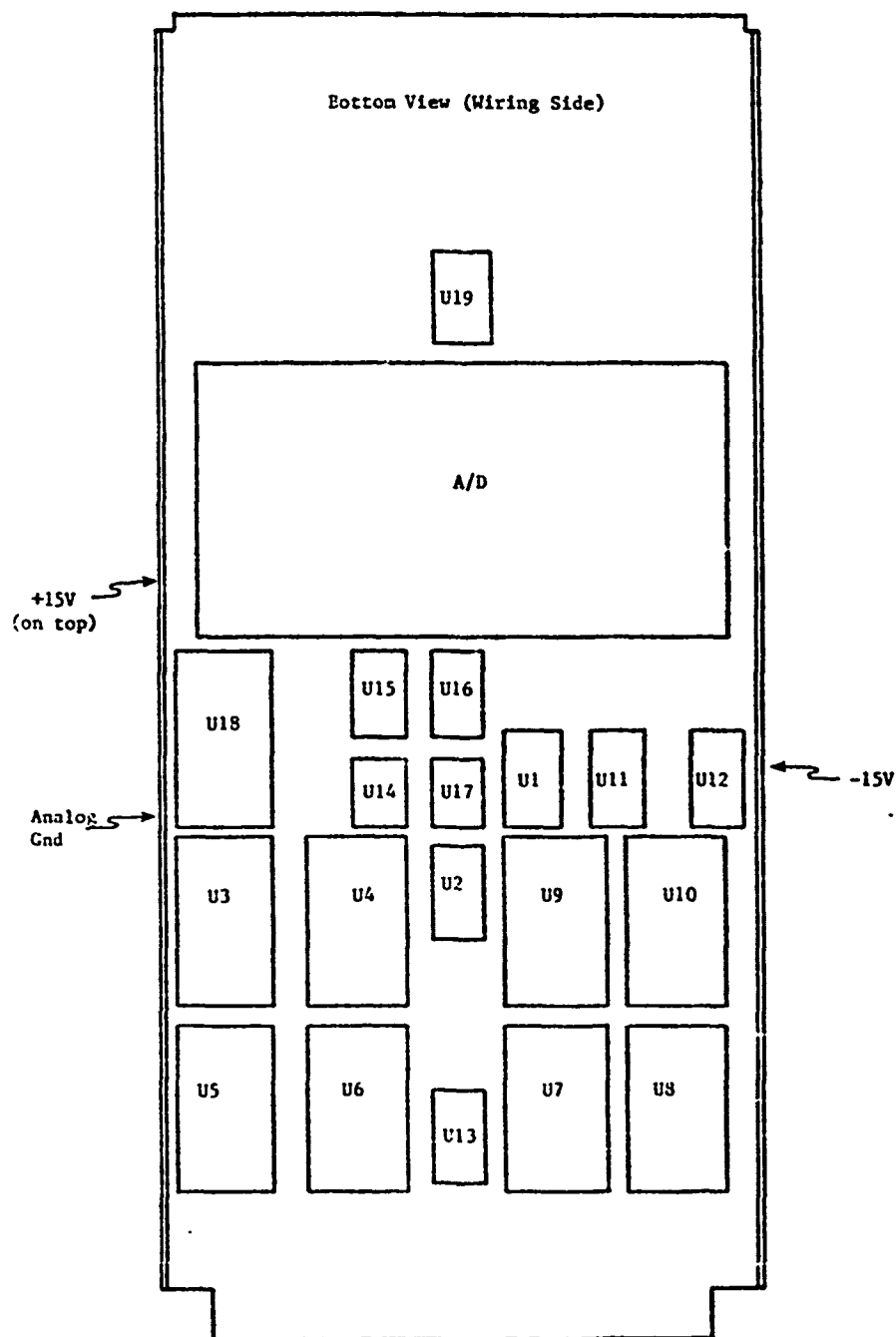
0039 ENDS





Appendix 2.5: Parts Layout for Video Interface

PARTS LAYOUT



DISTRIBUTION LIST

Copy No.

- 1 Lt. Commander Wayne Savage
Code 471, Rm. 704
Office of Naval Research
800 N. Quincy St.
Arlington, Va. 22217
- 2 Naval Post Graduate School
Monterey, CA 93940
Attn: Dr. T. F. Tao, Code 62TV
- 3 Naval Post Graduate School
Monterey, CA 93940
Attn: Library
- 4 - 7 Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217
Attn: ONR 240
- 8 Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217
Attn: ONR 250
- 9 Office of Naval Research
800 Quincy St.
Arlington, VA 22217
Attn: ONR 432
- 10 Naval Sea Systems Command
Washington, DC 20362
Attn: NAVSEA 0341
- 11 Naval Sea Systems Command
Washington, DC 20362
Attn: NAVSEA 03132
- 12 Naval Research Laboratory
Washington, DC 20375
Attn: NRL, Code 5550

DISTRIBUTION LIST (continued)

Copy No.

- 13 Naval Research Laboratory
 Washington, DC 20375
 Attn: NRL, Code 1409
- 14 Naval Surface Weapons Center
 Dahlgren Laboratory
 Dahlgren, VA 22448
 Attn: Code N-54
- 15 Naval Surface Weapons Center
 White Oak Laboratory
 Silver, Spring, MD 20910
 Attn: Code G-42
- 16 Naval Weapons Center
 China Lake, CA 93555
 Attn: NWC, Code 3945
- 17 Naval Weapons Center
 China Lake, CA 93555
 Attn: NWC, Code 3133
- 18 Naval Air Systems Command
 Washington, DC 20361
 Attn: NAVAIR 360
- 19 U. S. Army Electronics Command
 Fort Monmouth, NJ 07730
 Attn: Code NL-BP
- 20 - 31 Defense Documentation Center
 Bldg. 5, Cameron Station
 Alexandria, VA 22314
- 32 Defense Advanced Research Project Agency
 1400 Wilson Boulevard
 Arlington, VA 22209
 Attn: Dir. Target Acquisition Division

DISTRIBUTION LIST (continued)

Copy No.

33	Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709
34 - 35	E. S. McVey
36 - 38	E. A. Parrish
39 - 40	R. M. Inigo
41	I. A. Fischer Office of Sponsored Programs
42 - 43	E. H. Pancake Clark Hall
44	RLES Files

1139:rr

UNIVERSITY OF VIRGINIA

School of Engineering and Applied Science

The University of Virginia's School of Engineering and Applied Science has an undergraduate enrollment of approximately 1,400 students with a graduate enrollment of approximately 600. There are 125 faculty members, a majority of whom conduct research in addition to teaching.

Research is an integral part of the educational program and interests parallel academic specialties. These range from the classical engineering departments of Chemical, Civil, Electrical, and Mechanical and Aerospace to departments of Biomedical Engineering, Engineering Science and Systems, Materials Science, Nuclear Engineering and Engineering Physics, and Applied Mathematics and Computer Science. In addition to these departments, there are interdepartmental groups in the areas of Automatic Controls and Applied Mechanics. All departments offer the doctorate; the Biomedical and Materials Science Departments grant only graduate degrees.

The School of Engineering and Applied Science is an integral part of the University (approximately 1,530 full-time faculty with a total enrollment of about 16,000 full-time students), which also has professional schools of Architecture, Law, Medicine, Commerce, Business Administration, and Education. In addition, the College of Arts and Sciences houses departments of Mathematics, Physics, Chemistry and others relevant to the engineering research program. This University community provides opportunities for interdisciplinary work in pursuit of the basic goals of education, research, and public service.