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
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 potential 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.
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

November 1980
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>vii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
</tbody>
</table>

Chapter

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)  

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>51</td>
</tr>
<tr>
<td>61</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td>67</td>
</tr>
<tr>
<td>68</td>
</tr>
<tr>
<td>69</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>81</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS
(Continued)

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

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

Appendices

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

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Algorithm Comparison Chart</td>
</tr>
</tbody>
</table>
### LIST OF ILLUSTRATIONS

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

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>Multipliers and summers Circuit Diagram</td>
<td>55</td>
</tr>
<tr>
<td>3.11</td>
<td>SIPO Devices Circuit Diagram</td>
<td>56</td>
</tr>
<tr>
<td>3.12(a)</td>
<td>$\phi_1$ and $\phi_2$ waveforms 5V/cm; sweep at 0.5 ms/cm</td>
<td>58</td>
</tr>
<tr>
<td>3.12(b)</td>
<td>$\phi_c$ clock and output of PISO device No. 1</td>
<td>58</td>
</tr>
<tr>
<td>3.13(a)</td>
<td>PISO Devices Output</td>
<td>59</td>
</tr>
<tr>
<td>3.13(b)</td>
<td>First two Multipliers output</td>
<td>59</td>
</tr>
<tr>
<td>3.14(a)</td>
<td>Output of Summers</td>
<td>60</td>
</tr>
<tr>
<td>3.14(b)</td>
<td>SIPO Devices Output at taps No. 9</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>CCD Implementation of Widrow-Hoff Rule</td>
<td>69</td>
</tr>
<tr>
<td>4.2</td>
<td>Flowchart of Implementation of Widrow-Hoff Rule</td>
<td>72-73</td>
</tr>
<tr>
<td>4.3</td>
<td>Trans. Vector vs. # of Iterations (no dilation or rotation)</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>Trans. Vector vs. # of Iterations (no dilation or rotation)</td>
<td>76</td>
</tr>
<tr>
<td>4.5</td>
<td>Trans. Vector vs. # of Iterations (no dilation or rotation)</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>Trans. Vector vs. # of Iterations (no dilation or rotation)</td>
<td>78</td>
</tr>
<tr>
<td>4.7</td>
<td>Angle of Rotation vs. # of Iterations and Dilation vs. # of Iterations</td>
<td>80</td>
</tr>
<tr>
<td>4.8</td>
<td>Trans. Vector vs. # of Iterations w/Added Noise</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Example system configuration</td>
<td>107</td>
</tr>
<tr>
<td>5.2</td>
<td>Imager with data readout sequence</td>
<td>109</td>
</tr>
<tr>
<td>5.3</td>
<td>Preprocessor with normalization and storage</td>
<td>111</td>
</tr>
<tr>
<td>5.4</td>
<td>Relationship between coordinates and iteration index, k, of a 32 by 32 subimage</td>
<td>112</td>
</tr>
<tr>
<td>5.5</td>
<td>Moment estimator</td>
<td>114</td>
</tr>
<tr>
<td>5.6</td>
<td>Digital by analog multipliers for moment estimator</td>
<td>115</td>
</tr>
<tr>
<td>5.7</td>
<td>Moment estimator output</td>
<td>116</td>
</tr>
<tr>
<td>5.8</td>
<td>Digital Kalman parameter estimator</td>
<td>118</td>
</tr>
</tbody>
</table>
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 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. 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 credance 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:


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 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

<table>
<thead>
<tr>
<th>Data (Code)</th>
<th>Bit #</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Bit 12 is high</td>
</tr>
<tr>
<td>20000</td>
<td>Bit 13 is high</td>
</tr>
<tr>
<td>40000</td>
<td>Bit 14 is high</td>
</tr>
<tr>
<td>100000</td>
<td>Bit 15 is high</td>
</tr>
<tr>
<td>0</td>
<td>All bits are low</td>
</tr>
</tbody>
</table>

### Bits Used to Output the Data

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Motor Driver 1</th>
<th>Motor Driver 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Forward</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>Reverse</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>—</td>
<td>Forward</td>
</tr>
<tr>
<td>15</td>
<td>—</td>
<td>Reverse</td>
</tr>
</tbody>
</table>

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 CCN 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 TRANSFER

1. COMPUTER PROGRAM
   1. STC 12, C
      • • • •
      SERVICE SUBROUTINE

2. INTERFACE CARD
   SELECT CODE 12
   1. SET CONTROL
      2. CLEAR FLAG
      3. SET FLAG
      4. CLEAR CONTROL

3. INPUT DEVICE
   1. START
      2. DONE

   DATA
   CLEAR CONTROL

4. PROGRAMMER'S RESPONSIBILITY
   AUTOMATIC OPERATIONS

OUTPUT TRANSFER

1. COMPUTER PROGRAM
   1. OTA 13
      • • • •
      SERVICE SUBROUTINE

2. INTERFACE CARD
   SELECT CODE 13
   1. BUFFER
      2. DATA
      3. DATA
      4. DATA

3. OUTPUT DEVICE
   1. START
      2. DONE

   CLEAR CONTROL

4. PROGRAMMER'S RESPONSIBILITY
   AUTOMATIC OPERATIONS

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)

Pixel Clock (clock)

Frame Request

Line Clock (reset)

Column Counter

Frame Clock

Reset

Address data from H.P.

Comparator

Comparator

Starting Column Register

Ending Column Register

Start Conversion (to ADC)

Pixel Clock

Window data from H.P.

Comparator

Comparator

Starting Row Register

Ending Row Register

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 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:

<table>
<thead>
<tr>
<th>CODE</th>
<th>BIT 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSTART</td>
<td>0 1 0 X X X X X X X</td>
</tr>
<tr>
<td>XSTOP</td>
<td>0 0 0 X X X X X X X</td>
</tr>
<tr>
<td>YSTART</td>
<td>0 0 1 X X X X X X X</td>
</tr>
<tr>
<td>YSTOP</td>
<td>0 1 1 X X X X X X X</td>
</tr>
<tr>
<td>YSTOP with &quot;go&quot;</td>
<td>1 1 1 X X X X X X X</td>
</tr>
</tbody>
</table>

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. A brief development of the algorithm is given below (see ONR-CR-233-092-1 for complete details):

\[
\dot{\mathbf{a}} = D_c \ddot{\mathbf{a}}
\]

(2.1)

where

- \( \ddot{\mathbf{d}} \) is the N x 1 scene difference vector
- \( \dot{\mathbf{a}} \) is the 4 x 1 total affine parameter vector
- \( D_c \) is the constrained N x 4 matrix

\( D_c = [P_c : G] \)

(2.2)

where

- \( P_c \) is the N x 2 matrix of weighted spatial derivatives
- \( G \) is the N x 2 matrix of spatial derivatives

\[
D_c^+ = [P_c : G]^+ = \left[\begin{array}{c}
P_c^+ \quad P_c^+ GC^+ \end{array}\right]
\]

(2.3)

where

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

(2.4)

The affine vector can be estimated by

\[
\mathbf{a} = \left[\begin{array}{c}
\dot{a}_c \\
\dot{b}_c
\end{array}\right]
\]

(2.5)

where

\[
\dot{a}_c = (P_c^+ - P_c^+ GC^+) \ddot{d}
\]

(2.6)

\[
\dot{b}_c = C^+ \ddot{d}
\]
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 2.11 -- Open Loop Data (each point averaged ten times).
Figure 2.12 -- Open Loop Data (no averaging).
Figure 2.13: Open Loop Data (each point averaged 100 times).
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 2x2 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 x n matrix. For a 2 x 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 \( 2 \times 2 \) matrices

\[
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{bmatrix}
= \begin{bmatrix}
c_{11} & c_{12} \\
c_{21} & c_{22}
\end{bmatrix}
\]  

(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

\[
T = 2m+n-1 = 4+2-1 = 5
\]  

(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

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

\[
\begin{array}{ccccc}
  & t_1 & t_2 & t_3 & t_4 & t_5 \\
\text{OUT 1} & 0 & 0 & c_{11} & c_{12} & 0 \\
\text{OUT 2} & 0 & 0 & 0 & c_{21} & c_{22} \\
\end{array}
\]

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 $2 \times 2$ and $P_c^T$ is $2\times N$. For the subsystem of Fig. 3.2, we had

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

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

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

\[
\frac{V_{out}(z)}{V_{in}(z)} = H(z) = \sum_{n=0}^{N-1} h_n z^{-n} \quad (3.3.1.1)
\]

where

\[
h_n = \sum_{k=0}^{M-1} h_n^k 2^{-k} \quad (3.3.1.2)
\]
Figure 3.2 The implementation of \([P_c]^T\) 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

\[
M_x = \sum_{i=-m}^{m} \sum_{j=-n}^{n} p'(i,j)x_i
\]  

(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 \( x_i \)s can then be considered as weights of the image intensities \( p'(i,j) \). Formula (3.3.2.1) can be rewritten as

\[
M_x = \sum_{i=-m}^{m} p'(i,-n)x_i = \sum_{i=-m}^{m} p'(i,(-n+1))x_i + \cdots + \sum_{i=-m}^{m} p'(i,n)x_i
\]  

(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),

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

(3.3.2.3)

\[
M_x = \sum_{k=-n}^{n} M_k
\]  

(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 $M$. 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 $x$ 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/Analoc 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),

$$[P_c]^+ = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} P_{11} & P_{21} & \cdots & P_{1N} \\ P_{12} & P_{22} & \cdots & P_{2N} \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \end{bmatrix}$$

(3.3.3.1)

where

$$c_{11} = q_{11}P_{11} + P_{12}P_{12}$$

$$c_{12} = q_{11}P_{21} + q_{12}P_{22}$$

$$c_{21} = q_{12}P_{11} + q_{22}P_{12}$$

etc.
Figure 3.4 System for computing \( x = \sum_{i=-m}^{n} \prod_{j=-n}^{m} p(i,j)x_{i} \)
Defining
\[ P_{11} = \sum_{k=0}^{M-1} h_{11}^k 2^{-k} \]  \hspace{1cm} (3.3.2a)
\[ P_{12} = \sum_{k=0}^{M-1} h_{12}^k 2^{-k} \]  \hspace{1cm} (3.3.2b)
\[ P_{jm} = \sum_{k=0}^{M-1} h_{jm}^k 2^{-k} \] \hspace{1cm} (3.3.2c)
j = 1, 2 ; m = 1, 2, \ldots, N

the matrix multiplication (3.3.3.1) can be expressed in terms of the multiplication of the 2 x 2 matrix \([q]\) and the vectors
\[ P_j \triangleq \begin{bmatrix} P_{j1} \\ P_{j2} \end{bmatrix} \] \hspace{1cm} (3.3.3)

as follows,
\[ c_j \triangleq \begin{bmatrix} c_{1j} \\ c_{2j} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \begin{bmatrix} P_{j1} \\ P_{j2} \end{bmatrix} \] \hspace{1cm} (3.3.4)

Hence,
\[ [P_c]^+ = [c_1, c_2, c_3, \ldots, c_N] \] \hspace{1cm} (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 $p_j$ 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 $[Pc]$, $c_{jk}$, 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,

$$T_p = (ADC)_{time} + 2t$$

where $t$ is a clock pulse time.

b) Serial-parallel processor,

$$T_{sp} = (ADC)_{time} + 3t$$

(3.3.3.7)

c) All-serial processor,

$$T_s = N \cdot (ADC)_{time} + N \cdot 3t$$

$$= N \cdot T_{sp}$$

(3.3.3.8)

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/analog 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 $V_{\text{bias}}$ 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 $V_{DS}$_t for these two transistors will be equal to $V_{\text{in}1}$ 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

$$I_{D_3} = B \left( V_{\text{ref}} - V_T \right) V_{DS} - \frac{1}{2} V_{DS}^2$$

$$I_{D_4} = B \left( V_{\text{IN}2} - V_T \right) V_{DS} - \frac{1}{2} V_{DS}^2$$

The summation of $I_{D_4}$ and $-I_{D_3}$ is then

$$I_{D_4} - I_{D_3} = BV_{\text{in}1} (V_{\text{IN}2} - V_{\text{ref}})$$

If $V_{\text{ref}}$ is made equal to the dc bias level of $V_{\text{IN}2}$, we see from (3.4.3) that

$$I_{D_4} - I_{D_3} = BV_{\text{in}1} V_{\text{in}2}$$

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 [8] by performing a weighted summation of $I_{D_4}$ and $-I_{D_3}$. 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 $\mu$m, 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 ms/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, \( \varphi_1 \) and \( \varphi_2 \) which are the complement of each other and which work at the frequency of transfer, and \( \varphi_T \) which works at the loading frequency. One \( \varphi_T \) pulse is required for every 32 \( \varphi_2 \) 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 $\theta_T$ and $\theta_2$ are high, but $\theta_T$ must start at least 50 ns before and end at least 50 ns after the corresponding $\theta_2$ pulse. A combination logic TTL circuit was designed to obtain $\theta_T$, $\theta_1$, and $\theta_2$ from a single external clock signal. The signals are transferred entirely out of the CCD device during the next 32 clock pulses of $\theta_2$ and $\theta_1$, when $\theta_1$ is high and $\theta_2$ 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 $f > 2$ MHz. Consequently, NF was chosen to be 1 MHz, with $F = 31.25$ 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 Summer Circuit Diagram
Figure 3.11 SIPO Devices 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,

\[
V_1 = -mx \\
V_2 = -2 + mx
\]

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

\[
V_{M_1} + V_{M_2} = -m_1x - K + m_2x
\]

\[
= -K + (m_2 - m_1)x
\]

Waveforms for the experimental circuit are shown in Figs. 3.12 to 3.14. In Fig. 3.12(a), the upper trace shows the \(T\) clock pulse. The lower trace shows a few \(2\) clock pulses. The time relation between \(T\) and \(2\) 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 \(2\) high and \(1\) high, respectively. The output of the multipliers should be filtered in the final version of the processor, in order
Fig. 3.12(a) $\theta_1$ and $\theta_2$ waveforms 5V/cm; sweep at 0.5 µs/cm.

Fig. 3.12(b) $\theta_T$ 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 TSVIP 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 GHz. 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 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

\[ d = Da \]  

(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 \( 6 \times 1 \) 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

\[ e = Da - d \]  

(4.2.1.2)

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

\[ J(a) = ||Da - d|| \]

\[ = \sum_{k=1}^{N} (t^k d^k - d^k)^2 \]  

(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,
- \( a \) 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

\[
\mathbf{a}_{k+1} = \mathbf{a}_k - p_k \nabla J(\mathbf{a}_k)
\]

(4.2.1.4)

where

\( a_k \) and \( a_{k+1} \) represent the \( k^{th} \) and \( k+1^{th} \) 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

\[
J(a) = \sum_{k=1}^{N} 2(a^t D^k - d^t) = 2D^t (Da - d)
\]

(4.2.2.1)

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

\[
\mathbf{a}_{k+1} = \mathbf{a}_k - p_k D^t (Da_k - d)
\]

(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
Thus, this descent algorithm will determine a coefficient vector \( \mathbf{a} \) 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

\[
a_{k+1} = a_k + p_k (d_k - a_k^T D) D_k
\]

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,

$$(d^k - a_k b^k)$$

(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
\[
f(i,j) = \frac{\sin(i\Delta - 1)}{i\Delta} \frac{\sin(j\Delta/2.5)}{(j\Delta/2.5) + .3j\Delta+2}
\]

(4.3.1.1)

where \(\Delta\), 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 \(p_k = \frac{p_0}{k}\) 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.
Obtain \( J \) matrix, \( d \) vector, initial \( a \) vector, \( p_0 \), # of sign changes/iteration of \( k_1 \), and \( N \) (# of target points used).

Set \( k_1 = 1, k = 1, \) and \( p_k = p_0 \)

\[
a_{k+1} = a_k + p_k(d^k - a_k D^k)D^k
\]

Has sign of error changed?

Increment sign change counter.

Have desired # of sign changes occurred?

Yes

No

Reset sign change counter.

\[
\begin{align*}
    k_1 &= k_1 + 1 \\
    p_\ast &= p_0/k_1
\end{align*}
\]

Figure 4.2 Flowchart of Implementation of Sidrow-Joff 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 \( B(1) = B(2) = 0.5 \)
1. Starting pt. = 0,0; \( \alpha_0 = 0.1 \)
2. Starting pt. = 0,0; \( \alpha_1 = 0.1 \)

\# of sign changes/iteration = ?

Figure 4.3
Trans. Vector vs. # of iter. (no dilation or
derivation $B(1)=B(2)=0.03$ rotation)

1. Starting point $= 0.0 \text{, } v_0 = 0$
2. Starting point $= 0.7 \text{, } v_0 = 0.4$

$\#$ of sign changes/iter.$= 1$

Figure 4.4
Trans. Vector vs. # of Iterations (no dilation or rotation)

Perturbation B(1): B(2) = 1
1. Starting pt. = 0.0; \( x_0 = 1.4 
2. Starting pt. = 0.01, 0.01; \( x_0 = 0.95 

# of sign changes/iteration = 4
Trans. Vector vs. $\theta$ of Iter (no dilation or rotation)

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

1. Starting pt. 0,0, $\nu_0 = .5$  
2. Starting pt. .18, .18, $\nu_0 = .5$

$\theta$ 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.
Angle of Rotation vs. # of Iterations

\[ \alpha_{exact} = 60^\circ, \quad s = 25 \quad \# \text{ of sign changes/Iter.} = 5 \]

Dilation vs. # of Iterations

\[ \alpha_{exact} = 94^\circ, \quad s = 25 \quad \# \text{ of sign changes/Iter.} = 4 \]

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: $B(1) = 0.05$ (B(2), not plotted, yielded similar results)

starting point: .04

1. $\sigma_0 = 2$; noise $\sim N(0, 1)$
2. $\sigma_3 = 7$; noise $\sim N(0, 2)$
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
\[ y(k) = \sum_{i=0}^{N-1} a(k,i)x(k-i) \]  
(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 investi-
gation 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 process-
ing 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 $100 \times 100$ 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

\[
\frac{\partial f}{\partial t} = a_{11} \frac{\partial f}{\partial x} + a_{12} \frac{\partial f}{\partial y} + a_{21} \frac{\partial f}{\partial x} + a_{22} \frac{\partial f}{\partial y} + b_1 \frac{\partial f}{\partial x} + b_2 \frac{\partial f}{\partial y}
\]

(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 \( (a_{11}, a_{12}, a_{21}, a_{22}, b_1, \text{ and } b_2) \) 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,

\[ \frac{\partial f(x,y,t)}{\partial t} = f(x,y,t+T) - f(x,y,t) \]  

(5.2.1.2)

There is an interesting physical interpretation of the coefficients derived in these formulas, when the coefficients are constrained such that \( a_{21} = a_{22} = a_1 \) 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 \( \frac{d\theta}{dt} \) about some point \((x_o, y_o)\) and expanding at the rate of \( \frac{d}{dt} \) about this same point, which is moving with rectangular velocity components given by \( \frac{dx}{dt} \) and \( \frac{dy}{dt} \), then the constrained parameters give the following relationships:

\[ \frac{d\theta}{dt} = \tan^{-1}\left(\frac{a_2}{a_1}\right) \]  

(5.2.1.3a)

\[ 1 + \left(\frac{1}{\rho}\right) \frac{d\rho}{dt} = \sqrt{a_1^2 + a_2^2} \]  

(5.2.1.3b)

\[ \frac{dx}{dt} = b_1 \]  

(5.2.1.3c)

and

\[ \frac{dy}{dt} = b_2 \]  

(5.2.1.3d)

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

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

(5.2.1.4)

where

\[ a_1 = \left[\frac{\rho(t+T)}{\rho(t)}\right] \cos[\theta(t+T) - \theta(t)] \]

\[ a_2 = \left[\frac{\rho(t+T)}{\rho(t)}\right] \sin[\theta(t+T) - \theta(t)] \]

\[ b_1 = x_o(t+T) - x_o(t) \]

and

\[ b_2 = y_o(t+T) - y_o(t) \]
Thus, a relativistic set of parameters about changes in target orientation, position, and size between consecutive frames can be generated as:

\[ \Delta x = b_1 \]  
\[ \Delta y = b_2 \]  
\[ \Delta \theta = \tan^{-1} \left( \frac{a_2}{a_1} \right) \]  
and 
\[ \alpha = \sqrt{a_1^2 + a_2^2} \]

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 $a < x < b$ and $c < y < d$, then the integration operation on the fundamental gradient equation yields:

$$
\begin{align*}
& \int \int \left( \frac{\partial f}{\partial x} \right) dx \ dy = a_{11} \int \int x \left( \frac{\partial f}{\partial x} \right) dx \ dy + a_{12} \int \int y \left( \frac{\partial f}{\partial x} \right) dx \ dy \\
& \hspace{1cm} + b_1 \int \int \left( \frac{\partial f}{\partial x} \right) dx \ dy + a_{22} \int \int y \left( \frac{\partial f}{\partial y} \right) dx \ dy \\
& \hspace{1cm} + b_1 \int \int x \left( \frac{\partial f}{\partial y} \right) dx \ dy + a_{22} \int \int y \left( \frac{\partial f}{\partial y} \right) dx \ dy \\
& \hspace{1cm} + b_2 \int \int x \left( \frac{\partial f}{\partial y} \right) dx \ dy + b_2 \int \int y \left( \frac{\partial f}{\partial y} \right) dx \ dy
\end{align*}
$$

(5.2.2.1)

This equation can be further reduced by noting the following equalities of calculus:
\[
\frac{\partial f(x,y,t)}{\partial t} \int_a^b \int_c^d f(x,y,t) \, dx \, dy = \frac{\partial V(t)}{\partial t}
\]

(5.2.2.2a)

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

(5.2.2.2b)

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

(5.2.2.2c)

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

(5.2.2.2d)

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

(5.2.2.2e)

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

(5.2.2.2f)

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

(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

\[
\frac{\partial V(t)}{\partial t} = V(t+T) - V(t)
\]

(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 \( c \cdot N \) 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
\[ (x_1(t), y_1(t)), (x_2(t), y_2(t)), \ldots, (x_N(t), y_N(t)) \]

the linear set of equations become

\[
\begin{align*}
x(t+T) &= b_1 + a_{11}x_1(t) + a_{12}y_1(t) \\
y_1(t+T) &= b_2 + a_{21}x_1(t) + a_{22}y_1(t) \\
x_2(t+T) &= b_1 + a_{11}x_2(t) + a_{12}y_2(t) \\
y_2(t+T) &= b_2 + a_{21}x_2(t) + a_{22}y_2(t) \\
&\hspace{1cm} \ddots \hspace{1cm} \ddots \\
x_N(t+T) &= b_1 + a_{11}x_N(t) + a_{12}y_N(t) \\
y_N(t+T) &= b_2 + a_{21}x_N(t) + a_{22}y_N(t)
\end{align*}
\]

(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

\[ \mathbf{A}^T(k) = [a_{11}(k) \ a_{12}(k) \ a_{21}(k) \ a_{22}(k) \ b_1(k) \ b_2(k)] \]  

(5.2.4.1a)

and

\[ \mathbf{x}^T(k) = [x_1(k) \ x_2(k) \ x_3(k) \ x_4(k) \ x_5(k) \ x_6(k)] \]  

(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

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

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

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 $(p(k)X(k)X^T(k))$ 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 $6 \times 6$ 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

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

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

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

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

and

\[
f(r, \theta+\theta_0) \leftrightarrow F(w, \phi+\phi_0) \tag{5.2.5.2c}
\]

where \( f(r,\theta+\theta_0) \) represents a \( \theta_0 \) 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

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

Basicall1, 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 \( f(x,y,t) \leftrightarrow L(u,v,t) \) represents this transformation then the following properties result:
\[ f(ax, ay, t) \rightarrow L(u + \ln(a), v, t) \]  \hspace{1cm} (5.2.5.4a)

and

\[ f(r, \theta + \theta_0) \rightarrow L(u, v + \theta_0, t) \]  \hspace{1cm} (5.2.5.4b)

where \( f(r, \theta + \theta_0) \) represents a \( \theta_0 \) 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

$$\theta = \tan^{-1}\left[\frac{(Y^T - Y^B)}{(X^T - X^B)}\right]$$

(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 following 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
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

$$w(k,t+T) = w(k,t) = a_{11} x_1(k,t) + a_{22} x_2(k,t) + b_1 x_3(k,t) + b_2 x_4(k,t)$$

(5.3.3a)

where

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

(5.3.3b)
Figure 5.3
Preprocessor with normalization and storage

Diagram showing a preprocessor with normalization and storage. The diagram includes boxed components labeled with mathematical expressions and data flow arrows.
Figure 5.4
Relationship between coordinates and iteration index, k,
of a 32 by 32 subimage
\[
x_1(k,t) = \sum_{i=0}^{3} [(y+3)f(x+i, y+3, t) - (y)f(x+i, y, t)] \\
+ \sum_{j=0}^{3} [(x+3)f(x+j, y+3, t) - (x)f(x, y+j, t)] \\
-2\sum_{i=0}^{3} \sum_{j=0}^{3} f(x+i, y+j, t) \quad (5.3.3c)
\]

\[
x_2(k,t) = \sum_{i=0}^{3} (x+i)[f(x+i, y+3, t) - f(x+i, y, t)] \\
- \sum_{j=0}^{3} (y+j)[f(x+3, y+j, t) - f(x, y+j, t)] \quad (5.3.3d)
\]

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

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

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:

\[
A(k+1) = A(k) + V_A(k)X(k)[2(k) - A^T(k)X(k)]/\rho(k) \quad (5.3.4a)
\]

\[
V_A(k+1) = V_A(k) + V_u - V_A(k)X(k)A^T(k)V_A(k)/\rho(k) \quad (5.3.4b)
\]

where

\[
\rho(k) = V_V^2 + X^T(k)V_A(k)X(k) \quad (5.3.4c)
\]
Figure 5.5
Moment Estimator

16 DIGITAL BLOCK Alias
(See Figure 5-6)

Data

Clock

f(x,y)
f(x+1,y)
f(x+2,y)
f(x+3,y)
f(x,y+1)
f(x+1,y+1)
f(x+2,y+1)
f(x+3,y+1)
f(x,y+2)
f(x+1,y+2)
f(x+2,y+2)
f(x+3,y+2)
f(x,y+3)
f(x+1,y+3)
f(x+2,y+3)
f(x+3,y+3)

w(k)
x(k)
x(k)
x(k)
x(k)

OUTPUT

(See Figure 5-7)
Figure 5.6
Digital by analog multipliers for moment estimator
Figure 5.7
Moment Estimator Output
\[ A(k) = \begin{bmatrix} a_1(k,t) & a_2(k,t) & b_1(k,t) & b_2(k,t) \end{bmatrix} \] (5.3.4d)

\[ x(k) = \begin{bmatrix} x_1(k,t) & x_2(k,t) & x_3(k,t) & x_4(k,t) \end{bmatrix}^T \] (5.3.4e)

and

\[ Z(k) = W(k,t) - W(k,t-T) \] (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.
Computation Simplicity | GRADE | MONO | COORD | TRANS | SEGMENT
---|---|---|---|---|---
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 cannot 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


[16] Duda, R. O., and Hart, P. E., *Pattern Classification and Scene Analysis*


Using the Least Mean Square Gradient Algorithm," *IEEE Transactions on
pp. 494-507.

*IEEE Transactions on Communications*, Vol. Com-27, No. 2, Feb. 1979,
pp. 390-405.


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
Appendix 2.1: Pin Connections for Video Interface Cable

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>PIN</th>
<th>DEVICE END CONTINENTAL CONNECTOR PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT 0</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>BIT 1</td>
<td>B</td>
<td>E</td>
</tr>
<tr>
<td>BIT 2</td>
<td>C</td>
<td>K</td>
</tr>
<tr>
<td>BIT 3</td>
<td>D</td>
<td>P</td>
</tr>
<tr>
<td>BIT 4</td>
<td>E</td>
<td>U</td>
</tr>
<tr>
<td>BIT 5</td>
<td>F</td>
<td>Y</td>
</tr>
<tr>
<td>BIT 6</td>
<td>H</td>
<td>c</td>
</tr>
<tr>
<td>BIT 7</td>
<td>J</td>
<td>h</td>
</tr>
<tr>
<td>BIT 3</td>
<td>K</td>
<td>n</td>
</tr>
<tr>
<td>BIT 9</td>
<td>L</td>
<td>t</td>
</tr>
<tr>
<td>BIT 10</td>
<td>M</td>
<td>X</td>
</tr>
<tr>
<td>BIT 11</td>
<td>N</td>
<td>BB</td>
</tr>
<tr>
<td>BIT 12</td>
<td>P</td>
<td>FF</td>
</tr>
<tr>
<td>BIT 13</td>
<td>R</td>
<td>C</td>
</tr>
<tr>
<td>BIT 14</td>
<td>S</td>
<td>H</td>
</tr>
<tr>
<td>BIT 15</td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>DEVICE COMAND</td>
<td>22,Z</td>
<td>k,r</td>
</tr>
<tr>
<td>GROUND</td>
<td>BB</td>
<td>DO</td>
</tr>
</tbody>
</table>
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
The purpose of this routine is to provide the overall control for the tracking system.

This program calls the following routines:

- INIT - initialize control variables
- UTIL - read in data array and compute PC, G
- MEDIAN - median of scene 1 and 2
- TEST - input test variable
- GET - arrays of weighted and unweighted spatial derivatives
- GET - target vectors of scenes 1 & 2
- GET - array of affine estimates
- GET - maximum camera displacement from origin
- GET - number of averages for computed estimate
- GET - direction of displacement
- GET - noise adjustment factor
- GET - image data array

Best Available Copy
C STORE ARRAY NUMERALS
0062 DU$P(1)=IP$1
0063 DU$P(2)=IP$2
0064 DU$P(3)=IP$1
0065 DU$P(4)=JP$1
0066 C HUMAN TARGET CONTROL
0067 IF (I$OVE,NE,1) GO TO 490
0068 CALL UTIL(1, PIC, DUM$,D1, D2, "EAM1.G, PC")
0069 WRITE (1,109)
0070 IF ("ENTER 2 FOR SIMULATED AND 1 FOR REAL TARGET MOTION")
0071 READ (1,*), ITST
0072 IF (ITST.EQ,0) CALL PERT(PIC, IMAX, JMAX)
0074 IF (ITST.EQ,1) CALL DRIVE
0075 CALL UTIL(2, PIC, DUM$,D1, D2, "EAM1.G, PC")
0076 SCALE=EAM1/EAM2
0077 CALL TRACK(K, PC, G, SCALE, D1, D2, A)
0078 WRITE (1,108) POS1, POS2, A(3,1), A(4,1)
0079 CALL TST(Z, PC, A, POS1, POS2)
0080 IF (J$IS,NE,1) GO TO 700
0081 J$ER=0
0082 J$HO=1
0083 CALL UTIL(1, PIC, DUM$,D1, D2, "EAM1.G, PC")
0084 WRITE (1,109) JJ, A(3,1), A(4,1)
0085 CALL TRACK(K, PC, G, SCALE, D1, D2, A)
0086 WRITE (1,108) POS1, POS2
0087 CALL TST(Z, PC, A, POS1, POS2)
0088 CALL UTIL(2, PIC, DUM$,D1, D2, "EAM1.G, PC")
0089 CALL TRACK(K, PC, G, SCALE, D1, D2, A)
0090 WRITE (1,108) POS1, POS2
0091 CALL TST(Z, PC, A, POS1, POS2)
0092 IF (J$IS,NE,1) GO TO 700
0093 J$ER=0
0094 J$HO=1
0095 WRITE (1,108) POS1, POS2
0096 CALL TST(Z, PC, A, POS1, POS2)
0097 STOP
0098 C CLOSER LOOP OPERATION
0099 CALL UTIL(1, PIC, DUM$,D1, D2, "EAM1.G, PC")
0100 CALL UTIL(2, PIC, DUM$,D1, D2, "EAM2.G, PC")
0101 SCALE=EAM1/EAM2
0102 CALL TRACK(K, PC, G, SCALE, D1, D2, A)
0103 CALL PART(A(3,1), A(4,1))
** NO ERRORS**

PROGRAM = 19913  COMMON = A024

PAGE 1377 CONT  FORTRAN COMPILER: H24177 (SEPT. 1974)

ENDS
I.

THE PURPOSE OF THIS ROUTINE IS TO LOAD CONTROL CONSTANTS FOR THE INTERACTIVE TRACKING SOFTWARE PACKAGE. IF ANY NON-NUMERIC VALUE IS ENTERED FOR THE VARIOUS READ STATEMENTS THEN THE PROGRAM WILL DEFAULT TO THE GIVEN VALUES.

0016 CC SOURCE FILE: INIT1
0017 CC OBJECT FILE: INIT1
0018 CC CALLED BY: MAIN

0024 CC ISTART STARTING ADDRESS FOR CAMERA WINDOW
0021 CC ISTART+1 STARTING ADDRESS FOR CAMERA WINDOW
0023 CC ISTOP STARTING ADDRESS FOR CAMERA WINDOW
0024 CC JSTOP STARTING ADDRESS FOR CAMERA WINDOW
0027 CC ISTOP+1 STARTING ADDRESS FOR CAMERA WINDOW
0028 CC ISTOP+1+1 STARTING ADDRESS FOR CAMERA WINDOW
0029 CC ISTOP+1+2 SCENE DISPLAY OPTION
0030 CC DISPLAY DISPLAY EACH SCENE
0031 CC IDISP=2 DISPLAY DATA ARRAY
0032 CC IDISP=3 NO OUTPUT
0033 CC XAG "AGGREGATION FACTOR FOR DISPLAYED SCENE
0034 CC IX ADDRESS ON TECTRONIX DISPLAY
0035 CC JX ADDRESS ON TECTRONIX DISPLAY
0036 CC IVIRTV DERIVATIVE DISPLAY OPTION
0037 CC IVIRTV=1 DISPLAY DERIVATIVES
0038 CC IVIRTV=2 NO OUTPUT
0039 CC IIVIRTV DERIVATIVE COMPUTATION OPTION
0040 CC IIVIRTV=1 WITH AVERAGING
0041 CC IIVIRTV=2 NORMAL
0042 CC IIVIRTV=3 WITH MELON
0043 CC IIVIRTV=4 AUTOMATIC CONTROL OPTION
0044 CC IIVIRTV=5 CLOSED LOOP
0045 CC IIVIRTV=6 AUTOMATIC TARGET MOVEMENT
0046 CC IIVIRTV=7 HUMAN CONTROL
0047 CC IIVIRTV=8 NUMBER OF SAMPLES IN Y DIRECTION
0048 CC IIVIRTV=9 NUMBER OF SAMPLES IN Y DIRECTION
0049 CC IIVIRTV=10 TOTAL NUMBER OF SAMPLES
C

ISTART = 1

JSTART = 1

ISTOP = 99

JSTOP = 99

DISP = 3

PAGE = 2

I = 9

J = 9

IVIE = 1

IDOPT = 1

IDOVE = 1

IODEV = 1

WRITE (1, 110)

100 FORMAT ("ENTER D FOR DEFAULT", //)

WRITE (1, 201)

200 FORMAT ("ENTER ISTART, JSTART, ISTOP, JSTOP")

READ (1, *) ISTART, JSTART, ISTOP, JSTOP

WRITE (1, 301)

300 FORMAT ("ENTER DISPLAY OPTION: 0 NONE; 1 DISPLAY; 2 DUMP")

READ (1, *) IDISP

IF (IDISP .NE. 1) GO TO 400

IF (IDISP .NE. 1) GO TO 500

WRITE (1, 401)

400 FORMAT ("ENTER MAGNIFICATION AND COORDINATES OF SCENE")

READ (1, *) PMAC, IA, IB

500 IF (IDISP .NE. 2) GO TO 700

WRITE (1, 501)

501 READ (1, *) IPOS1, IPOS2, JPOS1, JPOS2

WRITE (1, 601)

601 FORMAT ("ENTER LOCATION OF DATA TO BE VIEWED")

READ (1, *) IPOS1, IPOS2, JPOS1, JPOS2

WRITE (1, 701)

701 READ (1, *) IVIE

801 FORMAT ("ENTER DECK DISPLAY OPTION: 0 NO DISPLAY; 1 DISPLAY")

READ (1, *) IVIE

WRITE (1, 901)

901 READ (1, *) II

WRITE (1, 1001)

1001 READ (1, *) IOPT

WRITE (1, 1101)

1101 READ (1, *) IOVE

WRITE (1, 1201)

1201 FORMAT ("ENTER OUTPUT DEVICE USED FOR AVERAGING")

READ (1, *) IODEV

WRITE (1, 1301)

1301 FORMAT ("ENTER CONTROL OPTION: 1 OPERATOR; 1 AUTOMATIC; 2 CLOSED LOOP")

READ (1, *) IOVE

WRITE (1, 1401)

1401 FORMAT ("ENTER OUTPUT DEVICE USED FOR AVERAGING")

READ (1, *) IODEV

WRITE (1, 1501)

1501 FORMAT ("ENTER OUTPUT DEVICE USED FOR AVERAGING")

READ (1, *) IODEV

WRITE (1, 1601)

1601 FORMAT ("ENTER OUTPUT DEVICE USED FOR AVERAGING")

READ (1, *) IODEV

WRITE (1, 1701)

1701 FORMAT ("ENTER OUTPUT DEVICE USED FOR AVERAGING")

READ (1, *) IODEV
0121  RETURN
0122  C
0123  C
0124  END

** NO ERRORS **
PROGRAM = 02465
COMMON = 03024

PAGE 0004 INIT FT14 COMPILED: HP24177 (SEPTEMBER 1974)
0125  ENDS
SUBROUTINE UTIL(IFLAG, PIC, N1, N2, MEAN, G, PORT)

THE PURPOSE OF THIS ROUTINE IS TO PERFORM GENERAL OUTPUT OPERATIONS IN THE TRACKING SOFTWARE PACKAGE.

SOURCE FILE: UTILS
OBJECT FILE: UTIL
CALLED BY: CONT

THIS PROGRAM CALLS THE FOLLOWING SUBROUTINES:

CAS READ DATA ARRAY FROM CAMERA
SORT REPACKS THE CAMERA DATA ARRAY
DCT PERFORMS TARGET / BACKGROUND SEPARATION
HOG STORES INTERIOR TARGET PTS TO CREATE D
ISV CREATE DERIVATIVE ARRAYS: PC & G
LISP OUTPUTS DIGITAL IMAGE OF TEXTON

IFLAG SCENE INDICATOR: 1 INITIAL, 2 SECOND
DI : 32 N X 1 ARRAYS CONTAINING TARGET POINTS
IRISP OUTPUT CONTROL VARIABLE: 1 DISPLAY IMAGE
PI ASC 2 X N SECTOR OF IMAGE; 3 DISPLAY IMAGE
PHI 3 X 1 ARRAY CONTAINING LOCATION OF
JOPOS TARGET PROJECTION ON X AXIS
JMPOS TARGET PROJECTION ON Y AXIS
I J ARRAY CONTROL VARIABLES
JOPOS ARRAYS OF WEIGHTED AND UNWEIGHTED
SPIRAL DERIVATIVES
FA REAL INTENSITY OF DATA ARRAY

INTER, ISTART, ISTOP, ISTOP, ISTART, IPAM, IFAIL, IPAM, IPOS, IPOS2
ITR 1, JSTOP, JPOS, JPOS2, ISEQ, ISEQ, IVIS, IVIS, ISEQ, ISEQ, ISEQ, ISEQ2
IIITR 1, J START, JSTOP, JPOS, JPOS2, ISEQ, ISEQ, ISEQ, ISEQ, ISEQ, ISEQ2
INTER, ISTART, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP, ISTOP
C SECOND SCENE: STORE TARGET POINTS
C IF (IFLAG, 0, 2) CALL DLOAD(PIC, 02)
C
C FIRST SCENE: SEGMENT; STORE TARGET POINTS; CALCULATE DERRIVATIVES
C IF (IFLAG, 1, 1) GO TO 100
C CALL ERECT(PIC)
C CALL DLOAD(PIC, 01)
C CALL DERM(PIC, 0, PC, x)
C
C LOAD SECTOR OF DATA ARRAY
C IF (1, MISP, 2, 0) GO TO 700
C WRITE (1, 200)
C WRITE (1, 200) IPOS1, IPOS2, JPOS1, JPOS2
C FORMAT (416)
C DO 609 I=DUMP(1), DUMP(2)
C DO 400 J=DUMP(3), DUMP(4)
C WRITE (1, 300) PIC(I, J)
C FORMAT (166)
C CONTINUE
C WRITE (1, 500)
C FORMAT (16, "##", 16)
C CONTINUE
C
C CALCULATE RANGE ON PIC(I, J) IN ORDER TO DISPLAY IMAGE
C IF (1, MISP, 1, 0) GO TO 500
C PHI=2PI
C PHI=1
C DO 300 I=1, IMAX
C DO 300 J=1, JMAX
C IF (PIC(I, J), LT, PHI) PHI=PIC(I, J)
C IF (PIC(I, J), GT, PHI) PHI=PIC(I, J)
C TARGET BLANKING
C IF (MISP, 1, 0) GO TO 500
C PHI=2
C IF (((I, J), LT, IPOS1), AND, (I, GT, IPOS2)) AND, ((J, LT, JPOS1), AND,
C + (J, GT, JPOS2)) PIC(I, J)=8
C CONTINUE
C CALL DIRE(PIC)
C CONTINUE
C RETURN
C END

** In G2028.P ** PROGRAM = "31" COMMON = "G28A

PAGE 0343 UTH. FOR A COMPILED: HP24177 (S. If., 1974)
SUBROUTINE SORT(PIC, XEAN)

THE PURPOSE OF THIS ROUTINE IS TO REPACK THE INTEGER DATA
ARRAY IN A MANNER WHICH PHYSICALLY RESEMBLES THE REAL WORLD.
THIS IS NECESSARY BECAUSE THE CAMERA'S DIGITAL DATA MAY NOT
MATCH THE FIXED FORTRAN ARRAY DIMENSIONS. THE ARRAY IS REPACKED
INTO ITSELF DUE TO THIS OPERATING SYSTEM'S MEMORY LIMITATIONS.

SOURCE FILE: SORS
OBJECT FILE: SORR
CALLED BY: MAIN

PIC INTEGER DATA ARRAY
IPAX NUMBER OF SAMPLES IN X DIMENSION
JAX NUMBER OF SAMPLES IN Y DIMENSION
ICOUNT TOTAL NUMBER OF SAMPLES
IPOS & JPOS POINTERS TO THE LOCATION OF NEXT ELEMENT
I & J DO LOOP CONTROL VARIABLES
HEAN REAL "HEAN"

INTEGER: ISTART, JSTART, ISTOP, JSTOP, ICOUNT, IPAX, JAX, IPOS1, IPOS2
INTEGER: JPOS1, JPOS2, IVEY, IDISP, IVIE, IYMT, IMVE, IPAG, IA, JA
INTEGER: ISTEP1, ISTEP2, IDR, POS1, POS2, POS1, POS2, POS1, POS2, POS1, POS2

REAL: "HEAN"

C

SET POINTS TO LAST ELEMENT IN EXISTING ARRAY
IPOS=ICOUNT/ICOUNT+1
JPOS=ICOUNT/ICOUNT+1 (IPOS-1)
IF NO. POINTERS EQUALS ZERO THEN RESET IT TO 1ST AND DECREMENT
COLUMN POINTER
IF (JPOS, IPOS) GO TO 1A
JPOS=IDR1
IPOS=IPOS-1
CONTINUE

HEAN REPACKING FROM LAST ELEMENT
DO 2A J=IPAX, 1, -1
2A
DO 2A I=IPAX, 1, -1

HEAN=HEAN+FLOAT(PIC(I,J)/ICOUNT)
PIC(I,J)=PIC(I,JPOS,IPOS)
DECREMENT ROW POINTER
JPOS=JPOS-1
0062 C IF ROW POINTER EQUALS ZERO THEN RESET IT TO 100 AND
0063 C DECREMENT THE COLUMN POINTER
0064 IF (JPOS,GT,100) GO TO 200
0065 JPOS=100
0066 IPOS=IPOS-1
0067 C
0068 200 CONTINUE
0069 C
0070 C RETURN
0071 C
0072 C
0073 C END

** NO ERRORS **
PROGRAM = 39124        COMMON = 3924
PAGE 0073 SORT FROM COMPILER: HP24177 (SEPT. 1974)
0074 C END
THE PURPOSE OF THIS ROUTINE IS TO PERFORM TARGET SELECTION
AND SEGMENTATION ON THE BASIS THAT THE TARGET INTENSITY DIFFERS
FROM THAT OF THE BACKGROUND. ROW AND COLUMN SUMS ARE CREATED
FROM THE INTEGER DATA ARRAY AND EACH ARE EXAMINED FOR THE TWO
GREATEST DIFFERENCES WHICH INDICATE THE POSITIONS WHERE THERE
IS THE GREATEST TRANSITION FROM BACKGROUND TO TARGET. WITH THESE
POSTIONS ON THE HORIZONTAL AND VERTICAL OF THE TARGET, 2D DATA
POINTS ARE SELECTED IN THE CENTER MOST PORTION OF THE ESTIMATED
TARGET REGION. THIS ALGORITHM ASSUMES A FAIRLY EVEN BACKGROUND
AND THAT THE TARGET HAS FAIRLY EQUAL DIMENSIONS.

SOURCE FILE: REGS
OBJECT FILE: REGS
CALLED BY: MAIN

THIS PROGRAM CALLS THE FOLLOWING ROUTINES:
ZER: TO INITIALIZE ARRAYS TO ZERO
SU: EXAMINE ROW AND COLUMN SUMS
LOG: TO ODER TARGET POSITION PARAMETERS AND
SELECT THE CENTER MOST TARGET REGION

PIC: INTEGER DATA ARRAY
IX: NUMBER OF SAMPLES IN THE X DIMENSION
JX: NUMBER OF SAMPLES IN THE Y DIMENSION
IY: A JSUB
JY: JSUB AND COLUMN SUMMING VECTORS
IXL1 < JYL2 TARGET BOUNDARIES IN X DIMENSION
JXL1 < JYR2 TARGET BOUNDARIES IN Y DIMENSION
I, J: LOOP CONTROL VARIABLES

INITIAL ARRAYS
CALL ZER (IX, JX, 1)
CALL ZER (JY, JY, 1)
C CREATE ROW SUM
3059 DO 1,2 I=1,IMAX
3060 DO 1,2 J=1, J'MAX
3061 ISU'(I)=ISU'(I)+PIC(I,J)
3062 1 CONTINUE
3063 C CREATE COLUMN SUM
3064 DO 2,1 J=1, J'MAX
3065 DO 2,1 I=1, I'MAX
3066 JSU'(J)=JSU'(J)+PIC(I,J)
3067 2 CONTINUE
3068 C EXAMINE ROW AND COLUMN VECTORS FOR THEIR TWO GREATEST DIFFERENCES
3070 CALL SU'(ISU',IMAX,IPOS1,IPOS2)
3071 CALL SU'(JSU',JMAX,JPOS1,JPOS2)
3072 C SELECT COORDINATES OF 20 INNERMOST DATA POINTS,
3073 CALL LOC
3074 C RETURN
3077 C RETURN
3078 C END
3080 ** NO ERRORS ** PROGRAM = U315 COMMON = P324
PAGE 1500 DIRECT FMCC COMPILER: HP24177 (SEPT. 1974)
C881 ENDS
SUBROUTINE SUM(VECT, POS1, POS2)

THE PURPOSE OF THIS ROUTINE IS TO EXAMINE THE VECTOR CONTAINING
THE SUMMATIONS AND DETERMINE THE LOCATIONS OF THE TWO LARGEST
DIFFERENCES BETWEEN THE SUMS VECT(1) AND VECT(1-1).

SOURCE FILE: SUMS
OBJECT FILE: SUMR
CALLED BY: RECT

VECT VECTOR CONTAINING SUMMATIONS
N NUMBER OF ELEMENTS IN THE VECTOR
POS1, POS2 POINTERS TO THE TWO GREATEST DIFFERENCES
GRA1, GRA2 VALUES OF THE TWO GREATEST DIFFERENCES
C CURRENT VALUE OF THE CURRENT DIFFERENCE
I DO LOOP CONTROL VARIABLE

IMPLICIT VECT(144), POS1, POS2, I, J, L, X, Y, Z, P, GRA1, GRA2, GRA3

C INITIALIZE POINTERS
POS1 = -9
POS2 = 1
C INITIALIZE GRADE 2 GRADE TO THE FIRST DIFFERENCE
GRA1 = IABS(VECT(1) - VECT(2))
GRA2 = GRA1
C EYE THE SUMMATING DIFFERENCES
DO 3 I = 3, 145
C GRADE = IABS(VECT(I) - VECT(I-1))
C IF CURRENT DIFFERENCE IS GREATER THAN LARGEST DIFFERENCE
C SAVE CURRENT DIFFERENCE AS NEXT GREATEST DIFFERENCE
C IF CURRENT DIFFERENCE IS NOT GREATER DIFFERENCE AND IS
C VALUE X WHERE ADJUST THE LOCATIONS OF THE DIFFERENCES
C BE (GRA2,GRA1,POS1,POS2) GO TO 100
GRA1 = GRA2
GRA2 = GRA1
POS1 = POS2
POS2 = I
GO TO 200
C
SUPPORT ROUTINE LOC

THE PURPOSE OF THIS ROUTINE IS TO ORDER THE TARGET COORDINATES IN EACH DIMENSION FROM LARGEST TO SMALLEST AND TO DETERMINE THE 2ND INTERMEDIATE DATA POINTS.

SOURCE FILE: LOC
OBJECT FILE: LOC
CALLED BY: RMT

IPOS 1, IPOS 2 TARGET BOUNDARIES IN X DIMENSION
JPOS 1, JPOS 2 TARGET BOUNDARIES IN Y DIMENSION
ITEMP, JTEMP TEMPORARY STORAGE VARIABLES
INUM, JNUM NUMBER OF SAMPLES IN X, Y DIMENSIONS
IPMON TOTAL NUMBER OF SAMPLES
SCALE SIZE ADJUSTMENT FACTOR

INTEGER ISTART, JSTART, ISTOP, JSTOP, COUNT, IAX, JAX, IPOS, JPOS
INTEGER JPOS1, JPOS2, ITEMP, JTEMP, IV,E, IMON, IP, IP, JSTART, JSTOP, JSTOP, COUNT, IAX, JAX, IPOS, JPOS
REAL SCALE

CO, COUNT ISTART, JSTART, ISTOP, JSTOP, COUNT, IAX, JAX, IPOS, JPOS?
CO, COUNT IPOS1, IPOS2, IPOS, JPOS1, JPOS2, JPOS?
CO, COUNT IPOS, JPOS1, JPOS2, IPOS, JPOS1, JPOS2, JPOS?
CO, COUNT ISTART, ISTOP, ISTOP, JSTOP, COUNT, IAX, JAX, JPOS, JPOS

OK, EQ IPOS1 AND IPOS2 (IPOS1=IPOS2)
IF (IPOS1, JPOS2) CO TO 1
ITEM=IPOS1
IPOS=IPOS2
IPOS2=ITEM

OK, EQ JPOS1 AND JPOS2 (JPOS1=JPOS2)
IF (JPOS1, JPOS2) CO TO 2
JTEMP=JPOS1
JPOS1=JPOS2
JPOS2=JTEMP

CALL CALCULATE MIDPOINTS
ITEM=(IPOS1+IPOS2)/2
ITEM=(JPOS1+JPOS2)/2

CALCULATE NUMBER OF POINTS IN EACH DIMENSION

ITEM=JPOS1-IPOS2+1
ITEM=JPOS2-JPOS1+1
IF (INO < 1) RETURN
SCALE=SQRT(INO/2)
IF (IPMOD < 2) GO TO 8
ADJUST NUMBER OF POINTS IN EACH DIMENSION BY SCALE FACTOR
INO=INT(INO/SCALE)
JNU=INT(JNU/SCALE)
CHECK SPECIAL CASES: INO=0 OR JNU=0
IF (INO<NE-N) GO TO 34
INO=1
JNU=2
35: IF (JNU<NE-N) GO TO 63
JNU=1
INO=2
COMPUTE NEW POSITIONS
IPOS1=ITEMP+INO/2
IPOS2=ITEMP-INO/2
JPOS1=ITEMP+JNU/2
JPOS2=ITEMP-JNU/2
RETURN
END
THE PURPOSE OF THIS ROUTINE IS TO CREATE A SCENE VECTOR THAT
CONTAINS THE POINTS WITHIN THE BOUNDARIES AS SPECIFIED BY
THE TARGET LOCATION IN THE INITIAL SCENE. THIS VECTOR WILL
LATER BE USED IN CALCULATING THE SCENE DIFFERENCE VECTOR.

SOURCE FILE:    MOAS
OBJECT FILE:    MOAR
CALLED BY:     "AI"

INTEGER DATA ARRAY

TARGET COORDINATES IN X DIMENSION
TARGET COORDINATES IN Y DIMENSION
VECTOR CONTAINING POINTS INTERIOR TO INITIAL SCENE
TARGET LOCATION
TOTAL NUMBER OF POINTS
NO LOOP CONTROL VARIABLES

INTEGER DATA POINTS IN A VECTOR

CONTINUE

THE
A-23

PAGE 2

STMA COMPILER: HP24177 (SEPT. 1974)

0011 C A-23.

0012 C

0013 C SOURCE FILE: DISP

0014 C

0015 C OBJECT FILE: DISP

0016 C

0017 C CALLED BY: MAIN

0018 C

0019 C PBE "MINIMUM INTEGER VALUE TO BE DISPLAYED"

0020 C MAX "MAXIMUM INTEGER VALUE TO BE DISPLAYED"

0021 C NAX "NUMBER OF SAMPLES IN X DIMENSION"

0022 C N "NUMBER OF SAMPLES IN Y DIMENSION"

0023 C P"SCALING FACTOR"

0024 C I" " " " " " " " " "" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " 
00259  J0  TO  I0  J=J0,J+,2
00260  JL=(MOD((J=J0-J+1),32)+96)*256
00261  J1=(J=J0-J+1)/32+32)*256
00262  IL=I=I1,I+,2
00263  J2=(MOD((I-1),32)+64)*256
00264  J3=(I-1)/32+32)*256
00265  IN=(I-I+1)/P*AG+1
00266  J4=(J-J+1)/P*AG+1
00267  IP=(I=I1,J=M)-P*IN+16/(P*MAX-P*IN+1)
00268  IF(I,GT,15) IP=15
00269  IF(I,LE,7) GOTO 1000
00370  WRITE (15,1) GRAY(IP),JH,JL,IH,IL.24329
00471  RETURN
00472  WRITE(15,1) 7936
00473  RETURN
00475  ENDS

** NO ERRORS**    PROGRAM = CA349    COMMON = CA349

PAGE 3073  DISP  FTRAN COMPILER: HP24177 (SEPT. 1974)
THE PURPOSE OF THIS ROUTINE IS TO CALCULATE THE WEIGHTED AND
WEIGHTED SPATIAL DERIVATIVES OF A PORTION OF AN IMAGE
DATA ARRAY FOR USE WITH THE INTERACTIVE VIDEO TRACKING SOFTWARE
SYSTEM.

SOURCE FILE: DERBS
OBJECT FILE: DERBS
CALLED BY: "MAIN"

INTEGER DATA ARRAY
TARGET BOUNDARIES IN X-DIMENSION
TARGET BOUNDARIES IN Y-DIMENSION
ARRAY OF UNWEIGHTED SPATIAL DERIVATIVES
ARRAY OF WEIGHTED SPATIAL DERIVATIVES
DERIVATIVE CALCULATION OPTION
DERIVATIVE DISPLAY OPTION
POINTERS USED IN DERIVATIVE CALCULATION
IMAGE PLANE ORIGIN COORDINATES
TOTAL NUMBER OF DERIVATIVES

TOTAL NUMBER OF POINTS IN THE ORGINAL IMAGE CAN BE ESTIMATED
AS (IPOS2+(IP0S1-IPOS2)/2), (IP0S2+(IP0S1-IPOS2)/2). THIS DEPENDS
ON A SPECIFICATION OF THE IMAGE PLANE COORDINATE SYSTEM. NOT THAT THE
"START" OF THE IMAGE ESTABLISHES A NEW IMAGE ORIGIN.

JPOS2=(JPOS1-JPOS2)/2
JPOS1=(JPOS2+1)-JPOS2/2

TOTAL NUMBER OF DERIVATIVES USED IN CALCULATIONS
RESIDUATIVE USING THE I+1 AND I-1 COORTS

1. I = I+1
2. I = I
3. I = I-1
4. j = I+1
5. j = I-1
6. I = 2
7. CONTINUE

IF (IV1, IV2, I, I) GO TO 700

OUTDER DERIVATIVE CALCULATIONS

IF (IV1, IV2, I, I) GO TO 700

X (I, J) = (X (I, J), J = 1, 2), I = 1, X

X (I, J) = (X (I, J), J = 1, 2), I = 1, X

Y (I, J) = (Y (I, J), J = 1, 2), I = 1, Y

Z (I, J) = (Z (I, J), J = 1, 2), I = 1, Z

GO TO 700

GO TO 700

GO TO 700

END

**Note:** The code snippet appears to be a FORTRAN program for computing derivatives using neighboring points. The comments and variable assignments suggest that it is part of a larger program, possibly for solving differential equations or another numerical method. The specific context or application is not clear from the code snippet alone.
SUBROUTINE T (X, Y, Z, SCALE, R, T, A)

THE PURPOSE OF THIS ROUTINE IS TO SOLVE THE TSVIP ALGORITHM FOR THE AFFINE PARAMETERS USING THE PSEUDO INVERSE (X = DOT * D)

SOURCE FILE: TRAC
OBJECT FILE: TRACR
CALLED BY: CONT

THIS PROGRAM CALLS THE FOLLOWING ROUTINES:
- PSU5L: CREATES THE PSEUDO INVERSE OF PC IN DOT
- TRA5: CREATES THE PRODUCT OF DOT AND D

A: 4 X 1 ARRAY CONTAINING THE AFFINE COEFFS
LC: ARRAY COMPOSED OF F AND G
U: ARRAY OF UNWEIGHTED SPATIAL DERIVATIVES
PC: ARRAY OF SPATIAL DERIVATIVES
K: NUMBER OF POINTS IN TARGET
J, J: LOOP CONTROL VARIABLES
SCALE: SCALE FACTOR TO REDUCE SENSITIVITY TO NOISE

SCALE = 0.5
SCALE = SCALE / SCALE

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)

CALL SUB (R2, R1, 0, 1)
SUBROUTINE PSFNU(A,APSFNU,"","",ATRN,"")

THE PURPOSE OF THIS ROUTINE IS TO CREATE THE PSFNU INVERSE
OF A AND STORE IT IN APSFNU. APSFNU = INVTRN(A)*INV(A)

SOURCE FILE: PSHS
OBJECT FILE: PI'W
CALLED BY: PAIN

THIS PROGRAM CALL THE FOLLOWING ROUTINES:
TUN
INV
MULTIPLES TWO ARRAYS
INV
PARAMETER ARRAYS: D1, D1, V
ATRN
TRANSPOSE OF ARRAY AT A yields V
APSFNU
PSFNU INVERSE OF ARRAY A OF DT, DT, DT
B
PRODUCT OF MATRICES ATRN AND D:
B
DUALLY STORAGE ARRAY: D1, D1, V
B & N
DIMENSIONS OF THE ARRAYS SPECIFIED ABOVE

INTEGER "",""
REAL A("",""),ATRN("",""),APSFNU("",""),B("","")

CREATE TRANSPOSE OF A "MATRX
CALL TRN(A,ATRN,"")
MULTIPLY THE TRANSPOSE OF A BY A AND STORE THE RESULT IN B
CALL "PY(ATERN,A,B,"")
STORE INVERSE OF B IN B
CALL INV(B,N,N,IFAIL)
IF INVERSE DOES NOT EXIST DETECT ERROR
WRITE("","")
FOR "NO-EXISTENT INVERSE"
STOP
MULTIPLY A BY THE TRANSPOSE OF A AND STORE RESULT IN APSFNU
CALL "PY(B,ATRN,APSFNU,"")
C
END

** NO ERRORS **  PROGRAM = PMA79  COMMON = PMA79

PAGE 3003  PSEUD  FTNA COMPILER: HP24177 (SEPT. 1974)

ENDS
SUBROUTINE INV(A,IA,N,IFAIL)

THE PURPOSE OF THIS ROUTINE IS TO FIND THE INVERSE OF A MATRIX BY GAUSS-JORDAN ELIMINATION, APPLYING TO THE ROWS OF THE IDENTITY MATRIX I THE SAME ELEMENTARY ROW TRANSFORMATIONS AS THOSE BY WHICH A IS REDUCED TO THE CANONICAL FORM I. THE INVERSE OF A IS OBTAINED. PARTIAL PIVOTING FOR THE LARGEST DIAGONAL ELEMENT IS ALSO DONE.

SOURCE FILE: INVS
OBJECT FILE: INV
CALLING BY: PSEU

A ALL N X N INPUT ARRAY WHOSE INVERSE IS DESIRED. NOTE A IS DESTROYED BY ITS INVERSE.
IA COLUMN LENGTH OF A
N THE ORDER OF THE SYSTEM
IOM TEMP STORAGE ARRAY OF I WORDS
TMP TEMP STORAGE ARRAY OF N WORDS

DIMENSION A(IA,IA),IOM(1,N),TEM(1,N),PI(5)

GO TO 14

14 IF(M1.LT.1) GO TO 40
30 DO 12=1,2
12 CONTINUE

GO TO 14

IF(A(L,L).EQ.0) GO TO 14

GO TO 40
LS=I+1.
A(LS,L)=A(LS,L)/A(L,L)
DO 14 I=1,K
LT=I+1.
IF (LT,GT,L) GO TO 14
LT=LT-1
14 A(LS,L,T)=A(LS,LT)+A(LS,L)*A(L,T)
IF (A(L,T),EO,'1) GO TO 13A
DO 64 I=1,K
A(L,L)=A(L,L)/A(L,T)
DO 32 I=1,K
LT=K-L+1.
LLL=LLL+1.
DO 24 J=1,LLL
LT=I-J+1.
TEMP(J)=A(L,J,LT)
A(L,J,LT)=A(J,LT)
IF (J,EO,LLL) A(L,J,LT)=A(J,LT)
32 CONTINUE
DO 3 J=I,L
DO 3 I=1,LLL
DO 3 I=1,L
DO 3 I=1,L
A(L',L)=A(L',L)-TEMP(')*A('2,L')
III=I+1.
A(L',III)=A(L',III)/TEMP(L+1)
DO 3 J=I,L
X=A(J,L')
A(J,L)=A(J,L')
DO 2 J=1,L
X=A(J,L')
CONTINUE
RETURN
10a IF (L.EQ.1) EXIT
RETURN
END
THE PURPOSE OF THIS ROUTINE IS TO SIMULATE TARGET MOTION BY

THE DATA ARRAY

SOURCE FILE: PERTS

OBJECT FILE: PERTP

CALLED BY: CONT

I IMAGE DATA ARRAY

DIMENSIONS OF DATA ARRAY

SHIFT FACTORS

LOCATION INDICES

INTEGER II (1,1,1)

WRITE (1,1)

FORMAT ("I.D. OUT HORIZONTAL AND VERTICAL PERTURBATIONS")

READ (1,*) ISHIFT, JSHIFT

DO 36 I=1,1

DO 36 J=1,1

IF (I

IF (I

II(I,J)=II(I,PTR,J)

CONTINUE

RETURN

END

** .10 ERRORS** PROGRAM = F0176 COMMON = FM100
THE PURPOSE OF THIS ROUTINE IS TO ALTER THE CAMERA POSITION
TO COMPENSATE FOR TARGET MOTION. THE SIGN OF THE AFFINE
ESTIMATES ARE EXAMINED TO DETERMINE THE DIRECTION OF THE MOTION

SUBROUTINE PANT(B1,B2)

SOURCE FILE: PANTS
OBJECT FILE: PANTIR
CALLED BY: CONT

THIS PROGRAM CALLS THE FOLLOWING Routines:

DRIVER FOR THE PAN TILT MOUNT
AFFINE ESTIMATES
CAMERA MOVEMENT
TEMPORARY STORAGE
PAN TILT SPEED CONTROL
PAN TILT DIRECTION CONTROL

IF (B1.50.0) JSSTEP1=0
IF (B2.50.0) JSSTEP2=0
I407 =3
B1=-B1
B2=-B2
IF ((B1.OE.1, AND)(B2.I.T.)) J1PR=2
IF ((B1.I.T., OR)(B2.I.T.)) J1PR=0
WRITE(1,1008) J1PR
CALL "DRIVER(12,JSSTEP1,JSSTEP2,1D14,IPER)"
POS1=POS1+513(JSTEP1,B1)
POS2=POS2+513(JSTEP2,B2)
RETURN
END
SUBROUTINE DRIVE

THE PURPOSE OF THIS ROUTINE IS TO ALLOW THE OPERATOR TO MOVE OR POSITION THE CAMERA SENSOR SOME SPECIFIED AMOUNT.

SOURCE FILE: DRIVES
OBJECT FILE: DRIVE
CALLED BY: CONT

THIS PROGRAM CALLS THE FOLLOWING ROUTINES:

STEP1 NUMBER OF MOTOR STEPS IN Y DIMENSION
STEP2 NUMBER OF MOTOR STEPS IN X DIMENSION
IDIR DIRECTION OF ROTATION
IFEP MOTOR SPEED: 1 LOW 2 MEDIUM 3 HIGH
POS1 & POS2 CURRENT CAMERA POSITION

INTEGER ISTART, JSTART, ISTOP, JSTOP, ICOUNT, IMAX, JMAX, IPOS1, IPOS2
INTEGER JPOS1, JPOS2, JOVE, JOPOS, JVIEW, JDOPT, JMOVE, PHAG, I, JA
INTEGER ISTEP1, ISTEP2, IDIR, POS1, POS2
COMMON ISTART, JSTART, ISTOP, JSTOP, ICOUNT, IMAX, JMAX, IPOS1, IPOS2
COMMON JPOS1, JPOS2, JOVE, JOPOS, JVIEW, JDOPT, JMOVE, PHAG, I, JA
COMMON ISTEP1, ISTEP2, IDIR, POS1, POS2

CALL DRIVER TO MOVE PAN TILT
CALL MOTOR(12, ISTEP1, ISTEP2, IDIR, IFEP)

CALCULATE NEW CAMERA POSITION

IF ((IDIR.EQ.0).OR.(IDIR.EQ.2)) ISTEP1=ISTEP1
IF ((IDIR.EQ.0).OR.(IDIR.EQ.1)) ISTEP2=ISTEP2
POS2 = POS2 + STEP2
C RETURN

** NO ERRORS **

PROGRAM = C3102
COMMON = C0024

PAGE 0603 DRIVE FTA4 CONTILER: HP24177 (SEP. 1974)

ENDS
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>0</td>
<td>CAS</td>
</tr>
<tr>
<td>000001</td>
<td>00</td>
<td>ENTR</td>
</tr>
<tr>
<td>000002</td>
<td>00</td>
<td>PAH</td>
</tr>
<tr>
<td>000003</td>
<td>00</td>
<td>ORBIT</td>
</tr>
<tr>
<td>000004</td>
<td>000000</td>
<td>IODEV</td>
</tr>
<tr>
<td>000005</td>
<td>000000</td>
<td>MASK</td>
</tr>
<tr>
<td>000006</td>
<td>000000</td>
<td>XSTRT</td>
</tr>
<tr>
<td>000007</td>
<td>000000</td>
<td>YSTOP</td>
</tr>
<tr>
<td>000008</td>
<td>000032</td>
<td>OUT</td>
</tr>
</tbody>
</table>

**NO ERRORS**
<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
PAGE 2024 - 01

1103 0016 120988  J P  **+9
1114 0016 070912  P1  OCT 1
1125 0016 090912  P2  OCT 2
1136 0016 030912  P3  OCT 3
1147 0016 070912  P4  OCT 4
1158 0016 090912  P5  OCT 5
1169 0016 030912  P6  OCT 6
1170 0016 070912  P7  OCT 7
1181 0016 090912  P8  OCT 8
1192 IS 0100 150912  IS  150912
1203 0016 070912  IST 1  IST 2
1214 0016 090912  IST 2  IST 3
1225 0016 070912  IST 3  IST 4
1236 0016 090912  IST 4  IST 5
1247 0016 070912  IST 5  IST 6
1258 0016 090912  IST 6  IST 7
1269 0016 070912  IST 7  IST 8
1270 0016 090912  IST 8  IST 9
1281 0016 070912  IST 9  IST 10
1292 0016 090912  IST 10  IST 11
1303 0016 070912  IST 11  IST 12
1314 0016 090912  IST 12  IST 13
1325 0016 070912  IST 13  IST 14
1336 0016 090912  IST 14  IST 15
1347 0016 070912  IST 15  IST 16
1358 0016 090912  IST 16  IST 17
1369 0016 070912  IST 17  IST 18
1370 0016 090912  IST 18  IST 19
1381 0016 070912  IST 19  IST 20
1392 0016 090912  IST 20  IST 21
1403 0016 070912  IST 21  IST 22
1414 0016 090912  IST 22  IST 23
1425 0016 070912  IST 23  IST 24
1436 0016 090912  IST 24  IST 25
1447 0016 070912  IST 25  IST 26
1458 0016 090912  IST 26  IST 27
1469 0016 070912  IST 27  IST 28
1470 0016 090912  IST 28  IST 29
1481 0016 070912  IST 29  IST 30
1492 0016 090912  IST 30  IST 31
1503 0016 070912  IST 31  IST 32
1514 0016 090912  IST 32  IST 33
1525 0016 070912  IST 33  IST 34
1536 0016 090912  IST 34  IST 35
1547 0016 070912  IST 35  IST 36
1558 0016 090912  IST 36  IST 37
1569 0016 070912  IST 37  IST 38
1570 0016 090912  IST 38  IST 39
1581 0016 070912  IST 39  IST 40
1592 0016 090912  IST 40  IST 41
1603 0016 070912  IST 41  IST 42
1614 0016 090912  IST 42  IST 43
1625 0016 070912  IST 43  IST 44
1636 0016 090912  IST 44  IST 45
1647 0016 070912  IST 45  IST 46
1658 0016 090912  IST 46  IST 47
1669 0016 070912  IST 47  IST 48
1670 0016 090912  IST 48  IST 49
1681 0016 070912  IST 49  IST 50
1692 0016 090912  IST 50  IST 51
1703 0016 070912  IST 51  IST 52
1714 0016 090912  IST 52  IST 53
1725 0016 070912  IST 53  IST 54
1736 0016 090912  IST 54  IST 55
1747 0016 070912  IST 55  IST 56
1758 0016 090912  IST 56  IST 57
1769 0016 070912  IST 57  IST 58
1770 0016 090912  IST 58  IST 59
1781 0016 070912  IST 59  IST 60
1792 0016 090912  IST 60  IST 61
1803 0016 070912  IST 61  IST 62
1814 0016 090912  IST 62  IST 63
1825 0016 070912  IST 63  IST 64
1836 0016 090912  IST 64  IST 65
1847 0016 070912  IST 65  IST 66
1858 0016 090912  IST 66  IST 67
1869 0016 070912  IST 67  IST 68
1870 0016 090912  IST 68  IST 69
1881 0016 070912  IST 69  IST 70
1892 0016 090912  IST 70  IST 71
1903 0016 070912  IST 71  IST 72
1914 0016 090912  IST 72  IST 73
1925 0016 070912  IST 73  IST 74
1936 0016 090912  IST 74  IST 75
1947 0016 070912  IST 75  IST 76
1958 0016 090912  IST 76  IST 77
1969 0016 070912  IST 77  IST 78
1970 0016 090912  IST 78  IST 79
1981 0016 070912  IST 79  IST 80
1992 0016 090912  IST 80  IST 81
2003 0016 070912  IST 81  IST 82
2014 0016 090912  IST 82  IST 83
2025 0016 070912  IST 83  IST 84
2036 0016 090912  IST 84  IST 85
2047 0016 070912  IST 85  IST 86
2058 0016 090912  IST 86  IST 87
2069 0016 070912  IST 87  IST 88
2070 0016 090912  IST 88  IST 89
2081 0016 070912  IST 89  IST 90
2092 0016 090912  IST 90  IST 91
2103 0016 070912  IST 91  IST 92
2114 0016 090912  IST 92  IST 93
2125 0016 070912  IST 93  IST 94
2136 0016 090912  IST 94  IST 95
2147 0016 070912  IST 95  IST 96
2158 0016 090912  IST 96  IST 97
2169 0016 070912  IST 97  IST 98
2170 0016 090912  IST 98  IST 99
2181 0016 070912  IST 99  IST 100
SUBROUTINE TP(A,B,C)

! THE PURPOSE OF THIS ROUTINE IS TO CREATE THE TRANSPOSE OF THE GIVN ARRAY (A) AND STORE THE RESULT IN THE OTHER ARRAY (B).

REAL A, B!

C REAL ARRAY: DIMENSION M X N: OPERAND
C B: REAL ARRAY: DIMENSION M X N: RESULT
C ! M, N: DIMENSIONS OF ARRAYS AS SPECIFIED ABOVE
C ! I: J: DO LOOP CONTROL VARIABLES

INTEGER I,J!

C
I = 1, L ("", ") R (", ")
J = 0, L (", ") R (", ")

C RETURN

END
THE PURPOSE OF THIS ROUTINE IS TO INITIALIZE A REAL ARRAY TO ZERO.

INTEGER I, J, N

REAL A(M, N)

DO 10 I = 1, M
    DO 20 J = 1, N
        A(I, J) = 0
    20 CONTINUE

RETURN
SUBROUTINE "PRY(A,B,C,M,N,P)"

THE PURPOSE OF THIS ROUTINE IS TO PERFORM MATRIX MULTIPLICATION
ON THE FIRST TWO ARRAYS AND STORE THE RESULT IN THE THIRD ARRAY
"MAT C = MAT A * MAT B".

THE REAL ARRAY: 1ST OPERAND: DIMENSION N X N
THE REAL ARRAY: 2ND OPERAND: DIMENSION N X P
THE REAL ARRAY: RESULT: DIMENSION N X P

DIMENSION OF ARRAYS SPECIFIED ABOVE

DO LOOP CONTROL VARIABLES

INTEGER I,J,K, M, P
REAL A(N,N), B(I,P), C(M,P)

CALL CLR(C,M,P)

PERFORM "MATRIX MULTIPLICATION"
DO 10 J=1,M
   DO 10 I=1,I
      C(I,J) = A(K,I)*B(K,J)
10 CONTINUE

END
SUBROUTINE ZER(A, M, N)

* THE PURPOSE OF THIS ROUTINE IS TO INITIALIZE AN INTEGER ARRAY *

TO ZERO.

SOURCE FILE: ZER

OBJECT FILE: ZER

CALLED BY: PECT

INTEGER A(M,N)

DIMENSIONS OF THE ARRAY

I, J

NO LOOP CONTROL VARIABLES

INTEGER 1, J, I, J

INTEGER A(1, 1)

LO 1 I=1,

LO 1 J=1,

A(I, J) = 0

CONTINUE

RETURN

END

FILE: ZER

COMMON = COMMON
SUBROUTINE SUB(A,B,C,"n")

THE PURPOSE OF THE ROUTINE IS TO GENERATE MATRIX SUBTRACTION

OF THE FORM C = A - B

SOURCE FILE: SUBS

OBJECT FILE: SUBR

CALLED BY: TRACK

A, B, & C

REAL ARRAYS WITH SUBTRACTION DEFINED AS

FOLLOWS C = A - B

M & I

DIMENSIONS OF THE ARRAYS

I & J

NO LOOP CONTROL VARIABLES

INTERSECT I, J, "n"

REAL A(I,"n"), B(I, J), C(1,"n")

C(1,J)=A(I,J)-B(I,J)

RETURN

END

** NO ERRORS** PROGRAM = "" COMMON = ""

PAGE 2432 SUB FTHA Compiler: UP24177 (Sept. 1974)
Appendix 2.4: Sequence of Segmented Tracking Scenes
Appendix 2.5: Parts Layout for Video Interface

PARTS LAYOUT

Bottom View (Wiring Side)

A/D

+15V (on top)

-15V

Analog

U18 U15 U16

U14 U17

U3 U4 U2

U9 U11 U12

U5 U6 U7 U8

U13
### DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Copy No.</th>
<th>Address Details</th>
</tr>
</thead>
</table>
| 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
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.