An adaptive multi-scene correlation algorithm

Paul E. Vogt
paul_v@bellsouth.net

Find similar works at: https://stars.library.ucf.edu/rtd
University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
https://stars.library.ucf.edu/rtd/4353
AN ADAPTIVE MULTI-SCENE CORRELATION ALGORITHM

BY

PAUL EDWIN VOGT
B.S.E., University of Central Florida, 1985
B.S.B.A., University of Florida, 1980

THESIS
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Fall Term
1988
ABSTRACT

Digital scene matching algorithms have been used in both military and commercial image processing systems for years. The trend toward using multiple sensors in military imaging systems has generated a new interest in real time techniques to accomplish sensor fusion tasks such as field of view alignment. This thesis analyzes methods presently in use and introduces a novel algorithm that improves scene correlation performance. The focus of the new technique is in the segmentation area, where significant features are extracted from background and clutter. These performance improvements are especially helpful when the scene contains excessive noise and or lacks detail, a trouble spot for standard correlation systems.

The restrictions imposed on the system design include implementations possible for real time processing and a minimum of hardware and power consumption. Simulations of the algorithms programmed for an image processing board hosted by an IBM personal computer are discussed.
ACKNOWLEDGEMENTS

I would like to express my appreciation to the Martin Marietta Corporation, specifically the Industrial Associates Program for enabling me to spend the time necessary to complete this thesis, and to Dr. Harley Myler and Mrs. Lynn Vining of the University of Central Florida for their guidance and support. I would also like to thank Teresa for her patience and understanding during the past two years.

COPYRIGHT (C) 1988 MARTIN MARIETTA CORPORATION
All rights reserved
This work is licensed to the University of Central Florida for academic purposes and Thesis Publication. This license also extends to software programs described in this document which were developed as part of this work.

iii
TABLE OF CONTENTS

LIST OF TABLES................................................................. V
LIST OF FIGURES............................................................... vi
INTRODUCTION................................................................. 1
VIDEO PREFILTERING............................................................. 7
FIELD OF VIEW EQUALIZATION............................................... 13
SEGMENTATION................................................................. 15
  Single-Bit Global Segmentation.......................................... 17
  Single-Bit Adaptive Segmentation ....................................... 18
  Two-Bit Global Segmentation............................................. 21
  Two-Bit Adaptive Segmentation.......................................... 23
CORRELATION....................................................................... 28
  Bilevel Correlation.......................................................... 30
  Two-Bit Correlation.......................................................... 31
  Modified Two-Bit Correlation.............................................. 33
ERROR PROCESSING.............................................................. 35
PERFORMANCE COMPARISON.................................................. 37
CORRELATOR IMPLEMENTATION............................................. 45
CONCLUSION.................................................................... 51
APPENDICES.................................................................. 53
  A. TURBO PASCAL SIMULATION PROGRAM............................... 54
  B. IMAGE PROCESSING SYSTEM........................................... 75
BIBLIOGRAPHY................................................................. 78
LIST OF TABLES

1. Field Of View Angular Subtense Correction ........ 14
2. Adaptive Two-Bit Algorithm Example ................. 25
3. Correlation vs. Convolution .......................... 29
4. Bilevel Correlation Truth Table ..................... 31
5. Normalized Two-Bit Correlation Truth Table ....... 32
6. Modified Two-Bit Correlation Truth Table ........... 33
7. Two-Bit Correlator Implementation Output Matrix ... 49
LIST OF FIGURES

1. Correlation System Block Diagram .................. 6
2. Photoconductor Noise Spectrum .......................... 9
3. Analog to Digital Conversion Relationship Illustrating Typical Error Sources ............... 10
4. Median Filter One- and Two-Dimensional Step Response ............................................. 11
5. Global Single-Bit Segmentation Applied to a Line of Composite Video .................................. 18
7. Global Two-Bit Segmentation Applied to a Line of Composite Video ..................................... 23
8. The Original Image Before Processing .................. 38
9. Global Single-Bit Segmentation Applied to the Image Shown in Figure 8 ...................... 39
10. Adaptive Single-Bit Segmentation Applied to the Image Shown in Figure 8 .................. 40
11. Global Two-Bit Segmentation Applied to the Image Shown in Figure 8 .......................... 41
12. Adaptive Two-Bit Segmentation Applied to the Image Shown in Figure 8 .......................... 42
13. Peak to Sidelobe Ratio Comparison for Five Independent Images ........................................ 44
14. Logic Devices CMOS Correlator ...................... 47
15. Two-Bit Correlator Implementation ............................ 48
16. Image Processing System Configuration .................. 76
INTRODUCTION

Different imaging sensor systems each have their advantages and disadvantages. For example, infrared imaging systems perform well under night and most inclement weather conditions except fog and heavy clouds, and are of a passive nature, giving them the advantage of nondetection. Millimeter wave radar has good performance in all types of weather, but has limited range and poor angle-to-angle resolution. Synthetic Aperture Radar systems have excellent poor weather performance but are predominantly side looking and require large amounts of power, weight, and area. Laser rangefinders also have good performance in weather but have a limited range and must be pointed very accurately at the target. Television type sensors have excellent range in good weather but are limited by darkness, fog, and clouds. By combining these sensors into one system, an optimum configuration is available for almost any weather, countermeasure, or tactical situation likely to be encountered [1]. However, there are many integration and configuration problems involved in obtaining optimum performance from this array of sensors. One such problem is aligning these sensors to insure they are all viewing
the same scene. We will define this as a multi-scene correlation problem. The solution to this problem shall comply with the following severe limitations of flyable hardware:

1) Real time operation.
2) Hardware size limitations.
3) Hardware power consumption limitations.

Real time operation is necessary to maintain proper alignment between the sensors. A large time constant between line of sight corrections would severely limit the effectiveness of a multiple sensor system. The ideal system will perform alignment every video frame. We are dealing with imaging sensors having 60 hertz frame rates, leaving 16.67 milliseconds for the image, error processing, and servo positioning operations.

The amount of hardware is also severely restricted in order to have minimum impact on size and weight. Aircraft designs have been optimized to provide for maximum performance by reducing these parameters. The goal of our system is to enhance performance, not to limit it by bulky electronics.

The last major restriction is on the power consumption of the electronics in the system. This is closely related to the size and weight limitation
mentioned above through the power supply required by the electronics. Every device utilized must have the minimum possible current draw. The advancement in Complimentary Metal Oxide Semiconductor (CMOS) technology has aided the designer in limiting power requirements.

There are two basic techniques available to accomplish the goal of image correlation:

1) Pixel level correlation.
2) Feature matching.

Pixel level correlation is the brute force method used to grab an area of interest in one image and spatially correlate it with the second image. The output of this operation is known as the correlation plane. This is actually a three-dimensional surface consisting of the x and y coordinates of the second image and an intensity representing the degree of correlation between the second image and the area of interest in the first image. The peak value of this surface is the location of the best match in the second image. By knowing where the area of interest is located in the first image and where the correlation peak is located in the second image, the relative offset between the two images can be calculated. These errors are then output to the sensor positioning
mechanism which slews the line of sight to minimize the alignment error.

The second method of aligning two images is feature matching. This technique involves extracting objects from the background clutter and calculating their position, possibly using the centroid of the object. By listing the objects and their positions in the field of view of both images, a feature matching algorithm can determine the relative offset of the images. For example, if image 1 has a boulder with centroid coordinates (128, 130) and image 2 has a similarly sized object with a centroid at (108, 160), then the relative offset can be calculated as (-20, +30). By comparing this offset with that of other features, an error can be output to the positioning mechanism.

Both of these algorithms depend on having a relatively large percentage of field of view overlap. This is not an unreasonable assumption as the sensors can be initially aligned at some point in time. Once this has occurred, the automatic correlation system can be utilized to maintain alignment.

The following sections illustrate and simulate the algorithms currently in use and introduce a new technique which improves correlation performance. We will use the
ratio of the peak value of the correlation surface to the average of the peak neighboring pixels as the performance index. This number is known as the Peak to Sidelobe Ratio (PSR) and is used as an indicator of the degree of scene match [2]. If the PSR is a large number, this indicates that the reference area has found a unique match in the second image. The correlation technique introduced in this thesis shows a substantial increase in the PSR. However, there is a costly penalty for this increased performance. The cost in this case is the increase in board space required for the new system. This is discussed in the implementation section, and possible techniques to minimize this cost are discussed.

The final section of a Multi-Scene Correlation System is the error processor. This subsection converts the correlation output plane into a voltage reference signal suitable for driving the sensor positioning system. It accomplishes this by scanning the correlation plane for the largest intensity value, corresponding to the location of the best spatial match between the two fields of view. The center of the correlation plane is known as the zero error point. The error signal is a relative value representing the difference between the peak location and the zero error point, initially
expressed as $x$ pixels by $y$ lines. This pixel by line error is converted into a voltage input to the servo system, driving the two fields of view into alignment. The block diagram for a single sensor and its signal processing system is shown in Figure 1.

![Figure 1. Correlation System Block Diagram.](image-url)
VIDEO PREFILTERING

The many different types of imaging sensors each have different noise susceptibility problems. For example, a FLIR system must minimize the following types of induced noise:

1) Photon noise due to the signal and background radiation.
2) Noise caused by the detector itself.
3) Noise induced by the signal processing electronics following the detector [3].

Photon noise is inherent in any type of electromagnetic radiation source. A Poisson distribution is the most accurate descriptor of the variability of the photon excitance. Over a period of time we can identify a mean value and variance for a given temperature, but we cannot accurately predict the number of photons.

The second major noise source in this type of system is detector induced. Johnson noise is caused by the thermal motion of electrons in a resistive element. Shot noise is due to the discrete nature of the photoelectrons generated by photons arriving at the detector.
Generation-Recombination noise in a photoconductor is caused by fluctuations in the generation and recombination of the current carriers. Another type of noise is described as l/f, due to the metallic contacts at the edge of the detector. This noise falls off rapidly with frequency, hence the l/f designation. Temperature noise is described as unwanted variations in the electrical output caused by detector temperature fluctuations not due to a change in the radiated signal. This is different than the Johnson noise associated with resistive elements mentioned above. Microphonic noise is caused by mechanical vibration of the components or wires which make up the system. This movement changes the capacitance of the wires with respect to ground and can actually modulate the detector output. Figure 2 illustrates the relative frequency spectrum of three major photoconductor noise sources.

The last major system noise input is created by the signal processing electronics themselves. The purpose of these electronics is to clean up the signal and place it in a format suitable for image processing. However, they introduce noise itself, such as Johnson noise, that is normally associated with any resistive element. Quantization is the process of converting the analog
Figure 2. Photoconductor Noise Spectrum.

detector output to a digital representation. This introduces quantization error, inherent in the analog to digital conversion, which is a roundoff or truncation error. This can be minimized by increasing the resolution or number of digital bits. Other errors include offset error, where the first transition does not occur at one half of the least significant bit; gain error, when the difference between where the values at which the first and last values occur is not equal; and linearity error, the differential voltage between transition values are not equal [4]. Figure 3 illustrates these errors for an analog ramp input.

As can be seen by the above examples, the signal output from a FLIR detector is corrupted with a variety of noise. The problem is the selection of an optimum filtering scheme. The goals of the filter should include:
Figure 3. Analog to Digital Conversion Relationship Illustrating Typical Error Sources.

1) Remove as much noise as possible without degrading the desired signal.

2) Increase the contrast (histogram stretching).

3) Preserve edge content.

Besides the conventional linear filters available to reduce the noise content of the video signal, a number of nonlinear filters exist which eliminate noise while allowing the high spatial frequency content of object edges to pass. This is an important feature to consider for optimum correlator performance. One such filter is known as the median filter [5]. The response of this
filter has the desirable feature that it does not roll off the edges of a step function input, but will eliminate a single impulse without affecting the surrounding pixels. The process of median filtering involves replacing the center pixel of an $N \times N$ window ($N$ odd) with the median value of the window. The shape

![One Dimensional Median Filter, N = 5](image)

![Two Dimensional Median Filter](image)

Figure 4. Median Filter One- and Two-Dimensional Step Response.
of the filter does not have to be square. A plus sign shaped filter, for example, will not roll off the corners of a square two-dimensional pulse. The size of the filter determines what size pulse will pass unaffected and what will be eliminated. The median filter pulse and step responses are shown in Figure 4. Another filter, the rank order filter, is a variation of the median, allowing replacement of the center pixel with the desired ranked element, such as the second highest value [6].

Another nonlinear type of filter is the out-of-range smoothing technique [7]. This filter is implemented by calculating the average in the N x N window, excluding the center pixel and subtracting the average from the center. If this difference is greater than a preset value, the pixel is replaced by the average; if not, the center pixel remains the same.

These nonlinear, digital, filtering techniques have the important property of eliminating random spikes without spreading the value over the adjacent pixels. This satisfies the third goal of the filtering system, preserving edge content.
FIELD OF VIEW EQUALIZATION

The most important task the preprocessing electronics must accomplish is field of view equalization. Virtually all types of sensors subtend different angles. Since we will be using a type of pattern or shape matching algorithm to accomplish our sensor alignment, the objects in the images must be scaled to the same size. This can be done by downsampling or averaging the pixels in one or both sensors involved in the correlation. This assists the overall image processing in two ways:

1) By averaging the pixels, a filtering operation can be included in the preprocessing electronics, thereby reducing space.

2) In most cases the video subtending the larger field of view is downsampled to equal the smaller field of view sensor output. Since the video line rate remains constant, this downsampling provides more processing time per pixel.

Table 1 illustrates an example angular subtense correction.
### TABLE 1
FIELD OF VIEW ANGULAR SUBTENSE CORRECTION

**INITIAL FOV SUBTENSE**

<table>
<thead>
<tr>
<th>FIELD OF VIEW (degrees)</th>
<th>PIXELS</th>
<th>DEGREES/PIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR 1</td>
<td>0.8</td>
<td>480</td>
</tr>
<tr>
<td>SENSOR 2</td>
<td>1.75</td>
<td>525</td>
</tr>
</tbody>
</table>

**TRANSFORM:**
- SENSOR 1: AVERAGE TWO PIXELS
- SENSOR 2: USE ONLY THE CENTER 240 PIXELS

**FINAL FOV SUBTENSE**

<table>
<thead>
<tr>
<th>FIELD OF VIEW (degrees)</th>
<th>PIXELS</th>
<th>DEGREES/PIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR 1</td>
<td>0.8</td>
<td>240</td>
</tr>
<tr>
<td>SENSOR 2</td>
<td>0.8</td>
<td>240</td>
</tr>
</tbody>
</table>

Since both sensors are viewing scenes at the same range, the images will now be scaled equivalently.
SEGMENTATION

Segmentation is the process of determining what pixels in the image belong to either the background or object category. Why is segmentation necessary in a multiscene correlation system? The computational requirements to do a correlation on a 256 x 256 x 8 bit image are too excessive to perform on real time, flyable hardware. Both time and frequency domain techniques are unsuitable if the data are left in their full eight-bit form. In order to eliminate this requirement and get suitable performance, segmentation is required. Specialized large scale integration (LSI) hardware has been developed to perform a binary correlation on 64 serial bit streams. A reference is preloaded into the chip and live data are clocked in. At the proper time, a correlation value representing the number of matching bits is output. The chip is simply performing a logical XNOR operation between the live data and the reference pattern. This specialized hardware saves space and power, making it ideal for a flyable application. The problem with this binary correlation is quantization error, similar to the analog to digital conversion
error. A possible way to minimize this error is by doubling the number of bits to two, providing four quantization levels. By using a two-bit thresholding scheme, the segmenter can assist in filtering low frequency noise which has passed through the video prefiltering subsection. Segmentation is an excellent way to isolate the true scene data for the correlation subsection. This thesis introduces a two-bit algorithm that provides the correlation subsystem with not only spatial information, but also relative intensity information. While this will not enable the same amount of discrimination as the full eight-bit data, it provides a compromise between eight- and single-bit correlation which satisfies both the performance and size goals of our flyable hardware.

Segmentation is accomplished by comparing the current pixel with a predetermined threshold. How to determine this threshold is crucial to the system success. Many different algorithms exist to compute this value. The output of the segmenter feeds the correlation subsystem of the image correlator. Determining which is the best segmentation algorithm involves knowing what input provides the best output from the correlator. The
correlator is basically a pattern matcher. In order to get the most discrimination out of the correlator, a pattern with the most detail is required. Correlator performance thrives on detail.

**Single-Bit Global Segmentation**

The simplest and oldest type of thresholding algorithm is the global threshold. Calculation of this threshold involves finding the maximum and minimum values in the image and dividing by two. This value is compared against every pixel in the image, and if the pixel is greater than the threshold, it is assigned a one, corresponding to an object. If not, it is assigned a zero and considered part of the scene background. This was the earliest and simplest algorithm utilized; however, it does not provide satisfactory detail. This type of threshold calculation also has the drawback that it uses either the previous field's peak values or delays the segmentation operation one field. The first case causes incorrect threshold calculation if there has been a significant change in the scene levels and the second case eliminates our goal of real time operation. Figure 5 illustrates the application of this type of thresholding to a typical line of EIA Standard composite video [8].
Figure 5. Global Single-Bit Segmentation Applied to a Line of Composite Video.

**Single-Bit Adaptive Segmentation**

An improvement to the basic threshold segmentation algorithm incorporates an adaptive threshold, based on the minimum and maximum values of different subregions of the screen. This overcomes the problem of using field old peak values to calculate the current threshold, but also introduces a delay into the binary video. The proper threshold cannot be calculated until the region is
completed. Other problems introduced include determining where to partition the subregions and in segmenting pixels in region border areas.

Improvement in segmentation performance can be obtained by calculating a single threshold for every pixel [9]. For a one-dimensional signal, such as a time dependent signal, an N pixel window is utilized: sum the N pixels located in the current window position and divide by N for an instantaneous average. If the center pixel in the window is greater than this average, then the pixel is assigned a 1 and considered an object. For a two-dimensional image, a two-dimensional window is necessary to shift around the image and create a unique threshold for each pixel. The size of the window or kernel is usually odd by odd in order to identify a center pixel. Each pixel value in the window is summed and divided by the number of pixels, to create a threshold for the center value. For example, a 9 x 9 window would be summed and divided by 81. This adaptive two-dimensional algorithm leads to a segmented image with the necessary detail to provide excellent correlation results. It also reduces the delay for thresholding the video to one half of the vertical filter size plus one half of the horizontal size. A problem with this adaptive
algorithm lies in how to threshold the pixels in the regions bordering the image. These pixels do not have a full kernel of values with which to calculate a proper threshold. Typically, the center of the image contains the information on which the correlation is to be performed, enabling the border pixels to be assigned a zero or background designation. Figure 6 illustrates an adaptive filter applied to a line of composite video where the center pixel of a nine pixel window is compared to the window average.

![Figure 6. Single-Bit Adaptive Filter of Size Nine Applied to a Line of Composite Video.](image-url)
Two-Bit Global Segmentation

The subject of this thesis takes the previous segmentation algorithms and adds another bit to the segmentation output. The result is a two-bit word instead of a single bilevel bit. A segmented pixel of value zero or one still represents scene background and object respectively; however, segmented pixels of values two and three represent objects of greater intensity. The advantage to doing this is that now the correlation subsystem has more information to use to discriminate between a match that is close and one that is exact. This new information can be considered as an intensity parameter. The technique can be applied to both algorithms introduced previously. In the case of global segmentation, the average value of the image can be calculated along with two other thresholds. The following algorithm can then be implemented:

1) If the pixel is less than or equal to the average(t1) value, it is assigned a zero.

2) If it is greater than the average(t1) and less than or equal to threshold2(t2), then it is assigned a one.

3) If it is greater than threshold2(t2) and less than or equal to threshold3(t3), then it is assigned a
two.

4) If the pixel value is greater than threshold3(t3), it is assigned a three.

We have increased the number of quantization levels from two to four by adding the additional bit to the segmentation output. The second two thresholds (t2 and t3) could be calculated in the following manner:

\[ t2 = \frac{(\text{peak} - \text{average})}{3} + \text{average} \]
\[ t3 = \frac{2 \times (\text{peak} - \text{average})}{3} + \text{average}. \]

For the following set of example data:

\[
\begin{align*}
\text{peak} & = 180 \\
\text{minimum} & = 120
\end{align*}
\]

the following thresholds would be calculated:

\[
\begin{align*}
\text{t1} & = \frac{(\text{peak} + \text{minimum})}{2} = \frac{180 + 120}{2} = 150. \\
\text{t2} & = \frac{(\text{peak} - \text{t1})}{3} + \text{t1} = \frac{180 - 150}{3} + 150 = 160. \\
\text{t3} & = 2 \times \frac{(\text{peak} - \text{t1})}{3} + \text{t1} = 2 \times \frac{180 - 150}{3} + 150 = 170.
\end{align*}
\]

Each pixel in the image would then be compared against the three thresholds and the result would be the segmented image. Using this type of algorithm to calculate the thresholds guarantees that the minimum and the maximum outputs will have at least one value for every scene. This is important because in order to benefit from the two-bit algorithm, pixels must fall in the different ranges. The rest of the pixels that have
intensity values between the maximum and minimum will fill in the other segmentation levels. Figure 7 illustrates this algorithm applied to a line of composite video.

![Graph illustrating intensity values and video pixels](image)

Figure 7. Global Two-Bit Segmentation Applied to a Line of Composite Video.

**Two-Bit Adaptive Segmentation**

While the two-bit global technique is an improvement over the global single-bit algorithm, the true value of quantizing to four levels is realized when applied to adaptive segmentation. Two-bit global segmentation retains the problem of peak latency while two-bit adaptive incorporates the benefits of single-bit adaptive
segmentation. This technique is the focus of this paper, combining adaptive thresholding with two-bit quantization. It retains the relatively minor problems of border pixel classification and a fixed quantization delay of one half of the vertical size plus one half of the horizontal size of the filter kernel, but the performance improvement is significant. An odd by odd number kernel is used to determine three unique thresholds for each pixel. The size of this kernel determines which frequencies will be detected or on which the threshold will ride. The larger the kernel, the more sluggish the threshold will be, causing lower frequencies to be detected. A small kernel will adapt quickly; low frequency information will not cross the threshold. Similar to the single-bit adaptive technique illustrated above, the average of the kernel is compared against the center pixel. A set of four ranges is calculated by multiplying constant factors against the current average. These two constant factors are critical to the success of the segmentation algorithm and depend on the response of the detector being used to generate the image. For the simulations performed in this thesis, 1.1 and 1.25 worked well to properly segment the image. These numbers were determined by analysis of the scene background and peak
levels; the relative thresholds were then applied to an image and histogram analysis verified the selection. When satisfactory levels of pixels were segmented into each range, the constant factors were fixed. The factors are multiplied against the average of the 9 x 9 kernel to calculate the second and third thresholds. After these thresholds are calculated, the pixel can be assigned a value using the same method as the two-bit global technique. The results of this type of segmentation provide excellent detail to the correlation subsystem. Table 2 shows algorithm execution.

**TABLE 2**

ADAPTIVE TWO-BIT ALGORITHM EXAMPLE

<table>
<thead>
<tr>
<th>CURRENT KERNEL:</th>
</tr>
</thead>
<tbody>
<tr>
<td>135 138 136 141 138 140 134 137 136</td>
</tr>
<tr>
<td>137 136 144 143 139 138 135 136 139</td>
</tr>
<tr>
<td>135 138 142 144 141 139 137 137 135</td>
</tr>
<tr>
<td>133 139 141 153 164 161 138 137 136</td>
</tr>
<tr>
<td>137 140 144 160 178 164 139 137 138</td>
</tr>
<tr>
<td>139 141 142 154 159 153 138 136 135</td>
</tr>
<tr>
<td>136 138 140 141 139 136 137 136 134</td>
</tr>
<tr>
<td>135 139 138 140 138 137 136 137 136</td>
</tr>
<tr>
<td>137 138 136 137 135 136 137 135 134</td>
</tr>
</tbody>
</table>
TABLE 2 -- CONTINUED

**THRESHOLD CALCULATIONS:**

9 x 9 average = \( \text{threshold}_1 = \frac{11,364}{81} = 140.3 \)

1.2 * \( \text{threshold}_1 = \text{threshold}_2 = 1.2 * 140.3 = 168.4 \)

1.5 * \( \text{threshold}_1 = \text{threshold}_3 = 1.5 * 140.3 = 210.5 \)

**OUTPUT ASSIGNMENTS:**

<table>
<thead>
<tr>
<th>range</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 140</td>
<td>0 0</td>
</tr>
<tr>
<td>141 - 168</td>
<td>0 1</td>
</tr>
<tr>
<td>169 - 210</td>
<td>1 0</td>
</tr>
<tr>
<td>211 - 255</td>
<td>1 1</td>
</tr>
</tbody>
</table>

**OUTPUT VALUE:**

178 => 1 0 (BINARY)

**OUTPUT KERNEL:**

```
0 0 0 1 0 0 0 0 0
0 0 1 1 0 0 0 0 0
0 0 1 1 1 0 0 0 0
0 0 1 1 1 1 0 0 0
0 0 1 1 2 1 0 0 0
0 1 1 1 1 1 0 0 0
0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```
The center pixel in this 9 x 9 kernel has an intensity value of 178, yielding a segmentation output of 2 (10 binary). In interpreting this value, it is an example of the extra piece of internal intensity information available from this algorithm. In contrast to the single bit method, this pixel would have been only labeled as a hot spot in the image. We now know that it is a hot spot within a hot spot, perhaps indicating the engine location in a jeep or other item of interest. Another advantage of this type of segmentation is its potential for application to other types of image processing, such as a target recognizer or target cuer. This additional internal intensity information could prove very valuable; for example, how many bushes or trees have intensity gradients internal to their shape?

Although this technique seems to involve many computations, all of the functions are easily realizeable with currently available digital signal processing hardware, including low power CMOS technology. This makes it ideal for the low power and minimum hardware implementation necessary for this application.
CORRELATION

Correlation is an operation used to identify where a pattern is located in an image [10]. It is usually utilized to locate where a relatively small area is located in a large image. The small area is shifted through every possible position in the large image and the spot of highest match is identified. The correlation is defined quantitatively as:

\[
R(m,n) = \sum_{x} \sum_{y} f(x,y)w(x-m,y-n).
\]

The maximum value of the correlation function \( R(m,n) \) is the position where \( w(x,y) \) matches \( f(x,y) \). From this equation, it is easily seen that for a 512 x 512 image, performing the correlation involves a substantial amount of calculations. At a 5 - 15 Mhz pixel processing rate, the hardware requirements for real-time operation become extensive. This is one reason the segmentation operation is used.

The correlation operation can also be carried out in the frequency domain. Since convolution in the time domain is equivalent to multiplication in the frequency domain and convolution is similar to correlation, it
makes sense that correlation can be done in the frequency domain. This is accomplished by taking the complex conjugate of one of the Fourier transformed images, prior to multiplication. This frequency domain technique also requires formidable computational requirements.

TABLE 3
CORRELATION VS. CONVOLUTION

<table>
<thead>
<tr>
<th>TIME DOMAIN</th>
<th>FREQUENCY DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CORRELATION</strong></td>
<td>( r(m,n) = \sum\sum f(x,y)w(x-m,y-n) )</td>
</tr>
<tr>
<td><strong>CONVOLUTION</strong></td>
<td>( c(m,n) = \sum\sum f(x,y)w(m-x,n-y) )</td>
</tr>
</tbody>
</table>

The procedure involves careful transformation of both images (they are usually different sizes), followed by a complex multiplication, and finally an inverse Fourier Transform. Even with the fastest signal processors available today and the implementation of the Fast Fourier Transform technique, a window larger that 16 x 16 is not suitable for real time [11]. The result of the
inverse Fourier Transform is the correlation plane. Simply find the maximum value and this is the point of best match. Besides the amount of time required to do the calculations, another major problem with this approach is the imperfect video that is output from the sensor. This problem is inherent in any frequency domain processing system, the different frequency responses and the noise susceptibility of transducers. The segmentation approach has a great benefit at this point, using a normalization technique to extract the objects from the noise and background.

**Bilevel Correlation**

Bilevel correlation is a special implementation of the time domain technique described above [12]. Instead of performing the multiplications in the equation, it is simply a summation of the number of pixels that match.

\[ R(m,n) = \sum \sum (f(x,y) \oplus w(x-m,y-n))' \]

where \( \oplus \) denotes the logical XOR operation. The correct result is achieved via the XNOR logical operation. The truth table is shown in Table 4.
TABLE 4
BILEVEL CORRELATION TRUTH TABLE

<table>
<thead>
<tr>
<th>f</th>
<th>w</th>
<th>(f (+) w)</th>
<th>(f (+) w)'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This is one reason the binary correlation operation is implemented in specialized hardware chips. By placing an array of these basic logical operators onto a silicon wafer, a large scale implementation can be achieved.

Two-Bit Correlation

Two-bit correlation is an extension of the bilevel case. If both bits match, the correlation output is the same as the bilevel case, a one. However, partial or close matches are now possible. Partial credit, one third, is awarded for a difference with an absolute value of one; two thirds is awarded for the case where the absolute value of the difference is two. This forces two corresponding pixels to be the same intensity
relative to their neighboring pixels in both sensors. This is where the two-bit correlator picks up its increase in discrimination capability. For the bilevel case, the pixels must simply be greater than their neighboring pixels to get a match. In the two-bit technique, the amount at which they are greater must fall into the same range to receive a perfect correlation score. The ideal correlation output values should fall according to the matrix shown in Table 5.

**TABLE 5**

NORMALIZED TWO-BIT CORRELATION TRUTH TABLE

<table>
<thead>
<tr>
<th></th>
<th>0 0</th>
<th>0 1</th>
<th>1 0</th>
<th>1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D 0 0</td>
<td>1.0</td>
<td>0.67</td>
<td>0.33</td>
<td>0.0</td>
</tr>
<tr>
<td>D 0 1</td>
<td>0.67</td>
<td>1.0</td>
<td>0.67</td>
<td>0.33</td>
</tr>
<tr>
<td>D 1 0</td>
<td>0.33</td>
<td>0.67</td>
<td>1.0</td>
<td>0.67</td>
</tr>
<tr>
<td>D 1 1</td>
<td>0.0</td>
<td>0.33</td>
<td>0.67</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Modified Two-Bit Correlation

The algorithm implemented in this thesis is a modification of this idealized correlation matrix. It does not give partial credit for reference and data that almost match. Only if the corresponding data are equal is the correlation sum incremented. This technique coupled with adaptive segmentation forces the data to be in the same intensity range relative to the surrounding pixels. This is very stringent criteria and provides for a more discriminating correlator. The output matrix is shown in Table 6.

<table>
<thead>
<tr>
<th>REFERENCE</th>
<th>0 0</th>
<th>0 1</th>
<th>1 0</th>
<th>1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D 0 0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A 0 1</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>T 1 0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>A 1 1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
This modified two-bit algorithm is simpler to simulate than the ideal case. Fractional sums disappear and there is no truncation or rounding error. The important consideration, however, is the performance improvement. This technique imposes a severe penalty (no partial credit) for pixels that are close in value but do not exactly match.
ERROR PROCESSING

Error processing is the final step of the multi-scene correlation system. It is a very straightforward function. The output of the system is an error signal to the sensor. This signal represents the amount of movement necessary to align the secondary or dependent sensor's line of sight to correspond with the main sensor's. The operation consists of:

1) Locating the peak value in the correlation plane.

2) Calculation of the relative offset between the area of interest in the first sensor and the peak.

The simulation of the algorithms took place using a 256 by 256 image and a relatively small reference window of sixteen pixels by sixteen lines. For this case the offset error would be calculated as follows:

\[ E(x,y) = (127 - \text{peak}(x) - 0.5Wx, \\
127 - \text{peak}(y) - 0.5Wy) \]

where peak(x) and peak(y) are the x and y coordinates of the peak correlation value, and 0.5Wx and 0.5Wy are the x and y half window sizes respectively. A movable reference window could be used with its center
coordinates replacing the (127,127) in the error equation.

For the simulations performed on the different segmentation and correlation techniques in this thesis, the error processing section output the peak value in the correlation plane, the peak to side lobe ratio, and the relative offset between the two images. The peak to side lobe ratio was used as a confidence coefficient in determining correlator performance.
PERFORMANCE COMPARISON

To evaluate any type of system performance, a quantitative measurement index must be used. In the case of a digital correlator, the index is the peak to sidelobe ratio. This measurement indicates how much the peak exceeds its nearest neighbors. This is a good indication of the peak's validity. A small value, approximately 1.0, indicates that the correlation system is unsure of the match, or that many good matches exist in this area. If we define the pixels as follows:

\[
\begin{align*}
&x-1,y-1 & x,y-1 & x+1,y-1 \\
&x-1,y & x,y & x+1,y \\
&x-1,y+1 & x,y+1 & x+1,y+1
\end{align*}
\]

then the peak to sidelobe ratio (psr) is:

\[
\text{psr} = \frac{R(x,y)}{R(x,y-1) + R(x-1,y) + R(x+1,y) + R(x,y+1)}/4,
\]

where \(R(a,b)\) is the correlation value at coordinates \((a,b)\). The method used to simulate the segmentation and correlation algorithms discussed above utilized the Weiland Image Processing System described in Appendix B. This system included a video camera and frame buffer able
to digitize a picture into memory. Figures 8 - 12 show the original image and the outputs of the four different segmentation techniques.

Figure 8. The Original Image Before Processing.
Figure 9. Global Single Bit-Segmentation Applied to the Image Shown in Figure 8.
Figure 10. Adaptive Single-Bit Segmentation Applied to the Image Shown in Figure 8.
Figure 11. Global Two-Bit Segmentation Applied to the Image Shown in Figure 8.
Figure 12. Adaptive Two-Bit Segmentation Applied to the Image Shown in Figure 8.
The images shown in these figures are representative of the database used to test the algorithms. They included varying signal to noise ratios and contrast levels. Note how the two global techniques in Figures 9 and 11 highlighted a significant amount of area which does not include objects. This is typical of global segmentation not following local averages. The adaptive single-bit segmentation exhibits a large amount of salt and pepper type noise. This is due to the low amplitude, high frequency content typical of a video signal. When this type of signal is compared to a low pass filtered version of itself, the high frequency information shows as transitions on the digitized output signal. The adaptive two-bit algorithm does the best segmentation function: separating the objects from the background. For this reason, this algorithm could be suitable for other image processing applications such as target recognition or cueing [13].

While the segmentation results are interesting, the goal of this system is the correlation of two images. Figure 13 plots a comparison of the Peak to Sidelobe Ratio results of the four different algorithms for five independent images.
Figure 13. Peak to Sidelobe Ratio Comparison for Five Independent Images.

Again the two-bit adaptive threshold proves superior to the alternate techniques. The difference between the single-bit and two-bit techniques is a function of how many pixels fall into the upper two quantization levels of the two-bit segmentations. This emphasizes the importance of setting optimum threshold levels. The other factor affecting this difference is the reference window size. For the simulations in this thesis, a relatively small reference of sixteen pixels by sixteen lines was utilized. A larger window will accentuate the advantage of two-bit quantization.
CORRELATOR IMPLEMENTATION

Implementation of correlation systems goes back as far as electronic systems themselves. The first correlators were a mass of analog comparators and summers. The problems of these systems were not only analog problems such as time and temperature drift, noise susceptibility, and component tolerances, but also a large amount of hardware and the resulting lack of precision. In the 1970's, signal processing component manufacturers introduced chips that assisted with some of the problems, but they were still analog implementations. For instance, TRW introduced a 64-bit binary correlator with a current output representing the degree of match of the bits [14]. By stacking x of these correlator chips, a correlation value could be determined in real time of a 64 pixel by x line array. This chip required a precision current reference for accurate correlation output.

As bipolar digital technology became more popular, the advantages of digital signal processing were immediately apparent for correlation systems. However, implementation of the logical XNOR operation for a large
image required huge amounts of hardware. Chip manufacturers responded to this by creating correlator chips similar to their analog predecessors, but outputting a digital word representing the correlation sum. Some of the problems with these chips are large amounts of current draw (bipolar versions) and the additional requirements of a large, fast adder tree to sum the x correlator outputs. In the previous analog versions, one op amp could sum a large number of correlator outputs. Now, a large number of digital adders had to be utilized and the time delay to sum these outputs slowed the system. CMOS technology has solved the power problem of the bipolar version of the digital correlators. Figure 14 shows a schematic of Logic Devices Inc.'s version of the CMOS correlator [15].
A two-bit digital correlator implementation exists which provides an exact (after scaling) match with the ideal correlation output using three of the digital correlator chips [16]. This implementation is shown in Figure 15.
Figure 15. Two-Bit Correlator Implementation.
This digital correlator configuration results in the correlation output matrix shown in Table 7. Notice that it is the two least significant bits which are not inputs to a common correlator chip.
This output matrix is identical (after scaling) to the ideal two-bit correlation function. Unfortunately, while this function is easily realized with currently available technology, the modified two-bit algorithm proposed in this thesis cannot utilize these digital correlator chips. It needs a slightly different function, logically ANDing the XNOR outputs on an individual pixel basis. While this is not a difficult function to implement, no LSI chips are currently available to realize it.

As the expertise in chip manufacturing technology increases, faster and more dense digital signal processors will become available. Not only will this enable the modified two-bit algorithm to be implemented, it will also enable more sophisticated segmentation and
correlation techniques to be realized within the constraints of a flyable environment.
CONCLUSION

Scene matching algorithms have existed for years; finding new ways to improve them is a formidable task. The approach used in this thesis is to minimize the quantization error in the data supplied to the correlation system, while complying with the restraints imposed on flyable hardware. This is coupled with an adaptive segmentation technique to maximize local contrast. The combination of these algorithms optimizes the pattern submitted to the correlation subsystem. The main function of the correlators is to match the detail from different real world scenes. This is only part of the information supplied by imaging sensors. The rest of the data includes atmospheric disturbances, sensor nonlinearities, and noise. The goal of segmentation is to reject this extraneous data and provide only scene detail information. By improving this data, the system performance can be increased.

This was proven by the performance of the two-bit adaptive algorithm on the images utilized by this thesis. The adaptive technique consistently outperformed global segmentation. For both global and adaptive, two-
bit quantization led to improved performance. By combining adaptive and two-bit, optimum correlation is achieved.

The two-bit adaptive segmenter introduced here has other applications besides the multi-scene correlator. Another immediate application is Automatic Target Recognition. The relative object intensity information would provide an excellent parameter to reject tank or jeep shaped bushes as clutter. Bushes typically do not have the inner hot spots included in the thermal profile of a jeep or tank.
APPENDIX A

TURBO PASCAL SIMULATION PROGRAM
TURBO PASCAL SIMULATION PROGRAM

This turbo pascal program receives as input a 256 x 256 eight-bit image and applies the algorithms described in this thesis to the input image. Output is provided at two places in the processing pipeline:

1) The output of the segmentation subsection.

2) The output of the error processing subsection.

The segmentation output is an image as illustrated in Figures 9, 10, 11, and 12. The output of the error processing subsection is the correlation peak, the peak to sidelobe ratio and the relative offset between the two images.
PROGRAM IMPROV;

{*******************************************************************************
*******************************************************************************
**
** THIS PROGRAM PERFORMS IMAGE PROCESSING
** ALGORITHMS ON IMAGES STORED ON THE WEILAND
** PCIP 100 BOARD. THE ALGORITHMS INCLUDE, BUT
** ARE NOT LIMITED TO THE FOLLOWING:
**
** 1. GLOBAL BILEVEL SLICING
** 2. ADAPTIVE BILEVEL SLICING
** 3. ADAPTIVE TWO-BIT SLICING
** 4. BILEVEL CORRELATION
** 5. TWO-BIT CORRELATION
** 6. CORRELATION PLANE THRESHOLDING
** 7. ERROR PROCESSING
**
** THE GOAL OF THIS SOFTWARE IS TO PROVIDE A
** QUANTITATIVE MEASURE OF THE RELATIVE OFFSET
** BETWEEN TWO FIELDS OF VIEW. THE OFFSET WILL
** BE EXPRESSED AS X PIXELS BY Y LINES.
**
*******************************************************************************
*******************************************************************************}

TYPE FILE_NAME = STRING[60];
FUNCTION IMCOPY(K,L,N,M:INTEGER):INTEGER;
EXTERNAL 'A:PCIPUTL.BIN';
FUNCTION IMMOVE(K,L,N,M:INTEGER):INTEGER;
EXTERNAL IMCOPY[3];
FUNCTION IMLOAD(VAR NAME:FILE_NAME;N,M:INTEGER):INTEGER;
EXTERNAL IMCOPY[6];
FUNCTION IMSAVE(VAR NAME:FILE_NAME;N,M:INTEGER) :INTEGER;
EXTERNAL IMCOPY[9];

VAR
I,J,K,L,M,N,Ll,L2 : INTEGER;
PIX1 : ARRAY[0..254] OF ARRAY[0..255] OF BYTE ABSOLUTE $8000:0;
PIX2 : ARRAY[0..254] OF ARRAY[0..255] OF BYTE ABSOLUTE $9000:0;
INP1,INP2,INP3,NAME1,NAME2,XX : FILE_NAME;
WINDOW : ARRAY[0..15] OF ARRAY[0..15] OF BYTE;
A,B,C,D : INTEGER;
CMAX,XCMAX,YCMAX,CDIFF,CT1,CT2 : INTEGER;
XERROR,YERROR : INTEGER;
XSUM11,YSUM11,XN11 : REAL;
XC11,XC12,XD : REAL;
XSUM12,YSUM12,XN12 : REAL;
YC11, YC12, YD : REAL;
PSR, SAV : REAL;
MDIFFX, MDIFFY : INTEGER;
MAXSUM1, MAXSUM2 : REAL;
XMAX1, XMAX2, YMAX1, YMAX2 : INTEGER;
MXS1, MXS2, XM1, XM2, YM1, YM2 : INTEGER;
DIFX, DIFY : INTEGER;
PROCEDURE SCRSET;

{******************************************************** **
** THIS PROCEDURE INITIALIZES THE SCREEN AND **
** PROMPTS THE USER FOR THE FILE OR FILES TO **
** BE SUBJECT TO THE IMAGE PROCESSING **
** ALGORITHMS SELECTED. **
********************************************************}

BEGIN
CLRSCR;
FOR I := 1 TO 5 DO WRITELN;
WRITELN('IMPROV: AN IMAGE PROCESSING ANALYSIS PROGRAM');
WRITELN; WRITELN;
WRITELN('ENTER THE NUMBER FOR OPERATION DESIRED: ');
WRITELN;
WRITELN('1. SEGMENTATION MENU ');
WRITELN;
WRITELN('2. CORRELATION MENU ');
WRITELN;
READLN(INP1);
CASE INP1 OF
1: BEGIN
CLRSCR;
FOR I := 1 TO 3 DO WRITELN;
WRITELN('SEGMENTATION MENU ')
WRITE('ENTER THE APPROPRIATE NUMBER FOR 
OPERATION DESIRED ');
WRITE('PICTURE FILE MUST BE LOADED INTO IMAGE 
PLANE 1,1 ');
WRITE('1. GLOBAL BILEVEL SEGMENTATION. ');
WRITE('2. GLOBAL 2 BIT SEGMENTATION. ');
WRITE('3. ADAPTIVE BILEVEL SEGMENTATION. ');
WRITE('4. ADAPTIVE 2 BIT SEGMENTATION. ');}
PROCEDURE GTHRESH;

{**************************************************************************
  THIS PROCEDURE CALCULATES A GLOBAL THRESHOLD FOR THE ENTIRE IMAGE
  AND DOES A THRESHOLDING FUNCTION TO PRODUCE BILEVEL (0 OR 1) VIDEO.
**************************************************************************}

VAR
GTHRESH,GMAX,GMIN : BYTE;
BEGIN
WRITELN;
WRITELN(' COMPUTING GLOBAL THRESHOLD AND PERFORMING SEGMENTATION. ');
WRITELN;
GMAX := 0;
GMIN := 255;
I := 0;
WHILE I < 255 DO
    BEGIN
        J := 0;
        WHILE J < 256 DO
            BEGIN
                IF PIX1[I,J] > GMAX THEN GMAX := PIX1[I,J];
                IF PIX1[I,J] < GMIN THEN GMIN := PIX1[I,J];
                J := J + 1;
            END;
        I := I + 1;
    END;
GTHRESH := (GMAX + GMIN) DIV 2;
I := 0;
WHILE I < 255 DO
    BEGIN
        J := 0;
        WHILE J < 256 DO
            BEGIN
                IF PIX1[I,J] > GTHRESH THEN PIX2[I,J] := 255
                ELSE PIX2[I,J] := 0;
                J := J + 1;
            END;
        I := I + 1;
    END;
END;

PROCEDURE GTWOBIT;
{******************************************************** **
** THIS PROCEDURE DOES A GLOBAL TWO BIT SEGMENTATION **
** ON THE IMAGE IN PLANE 1,1 AND PUTS THE RESULT IN **
** IMAGE PLANE 2,2. THE PROCEDURE IS SIMILAR TO THE **
** GTHRESH ABOVE WITH THE EXCEPTION THAT THE OUTPUT **
** PLANE IS QUANTIZED INTO TWO BITS INSTEAD OF ONE. **
********************************************************}
VAR
    TGMAX,TGMIN,TGTHRESH,TGTHRESHA,TGTHRESHB,TGTHRESHC : BYTE;
BEGIN
    WRITELN;
    WRITELN(' PERFORMING TWO BIT GLOBAL SEGMENTATION ON ');
    WRITELN;
    WRITELN(' THE IMAGE IN PLANE 1,1. ');
    WRITELN;
    TGMAX := 0;
    TGMIN := 255;
\[\begin{align*}
I & := 0; \\
\text{WHILE } I < 256 \text{ DO} & \begin{align*}
\text{BEGIN} & \\
J & := 0; \\
\text{WHILE } J < 256 \text{ DO} & \begin{align*}
\text{BEGIN} & \\
\text{IF } (\text{PIX1}[I,J] > \text{TGMAX}) \text{ THEN } \text{TGMAX} := \text{PIX1}[I,J]; & \\
\text{IF } (\text{PIX1}[I,J] < \text{TGMIN}) \text{ THEN } \text{TGMIN} := \text{PIX1}[I,J]; & \\
J & := J + 1; & \\
\text{END}; & \\
I & := I + 1; & \\
\text{END}; & \\
\text{TGTHRESH} & := (\text{TGMIN} + \text{TGMAX}) \text{ DIV 2}; & \\
\text{TGTHRESHA} & := (\text{TGTHRESH} + (\text{TGTHRESH} \text{ DIV 5})); & \\
\text{TGTHRESHB} & := (\text{TGTHRESH} + (\text{TGTHRESH} \text{ DIV 2})); & \\
I & := 0; \\
\text{WHILE } I < 256 \text{ DO} & \begin{align*}
\text{BEGIN} & \\
J & := 0; \\
\text{WHILE } J < 256 \text{ DO} & \begin{align*}
\text{BEGIN} & \\
\text{IF } (\text{PIX1}[I,J] \leq \text{TGTHRESH}) \text{ THEN } \text{PIX2}[I,J] := 0; & \\
\text{IF } (\text{PIX1}[I,J] > \text{TGTHRESH}) \text{ AND } (\text{PIX1}[I,J] \leq \text{TGTHRESHA}) \text{ THEN } \text{PIX2}[I,J] := 85; & \\
\text{IF } (\text{PIX1}[I,J] > \text{TGTHRESHA}) \text{ AND } (\text{PIX1}[I,J] \leq \text{TGTHRESHB}) \text{ THEN } \text{PIX2}[I,J] := 170; & \\
\text{IF } (\text{PIX1}[I,J] > \text{TGTHRESHB}) \text{ THEN } \text{PIX2}[I,J] := 255; & \\
J & := J + 1; & \\
\text{END}; & \\
I & := I + 1; & \\
\text{END}; & \\
\text{END}; & \\
\text{END}; & \\
\text{PROCEDURE} \text{ ATHRESH}; & \\
\{ & \\
\text{******************************************************} & \\
\text{THIS PROCEDURE DOES AN ADAPTIVE THRESHOLDING} & \\
\text{OPERATION ON THE PICTURE BY CALCULATING A} & \\
\text{LOCAL AVERAGE OF THE 9 X 9 KERNEL SURROUNDING} & \\
\text{THE CENTER PIXEL. IF THE CENTER PIXEL IS} & \\
\text{GREATER THAN THE LOCAL AVERAGE, THEN THE} & \\
\text{PIXEL IS ASSIGNED A 1, IF NOT IT IS ASSIGNED} & \\
\text{A 0.} & \\
\text{******************************************************}\} & \\
\end{align*}
\end{align*}
\end{align*}
\]
VAR
ATHRESH : INTEGER;
MA,NA,MB,NB : INTEGER;

BEGIN
WRITELN;
WRITELN(' COMPUTING ADAPTIVE THRESHOLDS AND PERFORMING SEGMENTATION. ');
WRITELN;
I := 0;
WHILE I < 255 DO
BEGIN
J := 0;
WHILE J < 256 DO
BEGIN
ATHRESH := 0;
FOR MA := (I - 4) TO (I + 4) DO
FOR NA := (J - 4) TO (J + 4) DO
BEGIN
MB := MA;
NB := NA;
IF MB < 0 THEN MB := 0;
IF MB > 255 THEN MB := 255;
IF NB < 0 THEN NB := 0;
IF NB > 255 THEN NB := 255;
ATHRESH := ATHRESH + PIX1[MB,NB];
END;
ATHRESH := ATHRESH DIV 81;
IF PIX1[I,J] > ATHRESH THEN PIX2[I,J] := 255
ELSE PIX2[I,J] := 0;
J := J + 1;
END;
I := I + 1;
END;
END;

PROCEDURE ATWOBIT;

{*******************************************************************************
** THIS PROCEDURE DOES THE MUCH ANTICIPATED TWO BIT THRESHOLDING OPERATION. IT IS MUCH LIKE **
** THE ATHRESH ROUTINE WITH THE EXCEPTION THAT **
** THE OUTPUT RANGE IS FROM 0 TO 3 INSTEAD OF **
** A 0 OR 1. THIS RESULTS IN A "STEEPER" **
** CORRELATION SURFACE THAN THE ONE BIT OPERATION. **
*******************************************************************************}
VAR
TWOTHRESH,TWOTHRESHA,TWOTHRESHB,TWOTHRESHC : INTEGER;
MC,MD,NC,ND : INTEGER;
BEGIN
WRITELN;
WRITELN(' COMPUTING TWO BIT THRESHOLDS AND PERFORMING SEGMENTATION. ');
WRITELN;
I := 0;
WHILE I < 255 DO
BEGIN
J := 0;
WHILE J < 256 DO
BEGIN
TWOTHRESH := 0;
FOR MC := (I - 4) TO (I + 4) DO
FOR NC := (J - 4) TO (J + 4) DO
BEGIN
MD := MC;
ND := NC;
IF MD < 0 THEN MD := 0;
IF MD > 255 THEN MD := 255;
IF ND < 0 THEN ND := 0;
IF ND > 255 THEN ND := 255;
TWOTHRESH := TWOTHRESH + PIX1[MD,ND];
END;
TWOTHRESH := TWOTHRESH DIV 81;
TWOTHRESHA := TWOTHRESH + (TWOTHRESH DIV 10);
TWOTHRESHB := TWOTHRESH + (TWOTHRESH DIV 4);
IF (PIX1[I,J] <= TWOTHRESH) THEN PIX2[I,J] := 0;
IF (PIX1[I,J] > TWOTHRESHB) THEN PIX2[I,J] := 255;
J := J + 1;
END;
I := I + 1;
END;
END;
PROCEDURE LOADREF;

{****************************************************** **
** THIS PROCEDURE LOADS THE CORRELATION WINDOW **
** WITH THE CENTER 16 X 16 PIXELS OF IMAGE PIXL **
******************************************************}

BEGIN
A := 0;
I := 119;

{** LOAD REFERENCE WINDOW **}

WHILE A < 16 DO
BEGIN
B := 0;
J := 119;
WHILE B < 16 DO
BEGIN
WINDOW[A,B] := PIXL[I,J];
B := B + 1;
J := J + 1;
END;
A := A + 1;
I := I + 1;
END;

{* CLEAR SCREEN WHERE THE CORRELATION PLANE IS NOT *}

FOR C := 0 TO 91 DO
FOR D := 0 TO 255 DO PIXL[C,D] := 0;
FOR C := 91 TO 148 DO
BEGIN
FOR D := 0 TO 91 DO PIXL[C,D] := 0;
FOR D := 148 TO 255 DO PIXL[C,D] := 0;
END;
FOR C := 148 TO 255 DO
BEGIN
FOR D := 0 TO 255 DO PIXL[C,D] := 0;
END;
PROCEDURE OUTERROR;

{*******************************************************************************
** THIS PROCEDURE OUTPUTS THE RESULTS OF THE
** CORRELATION OPERATION IN TABULAR FORM
** THESE RESULTS ARE QUANTIZED IN TERMS OF THE
** DIFFERENCE BETWEEN THE CORRELATION PEAK
** LOCATION AND THE CENTER OF THE IMAGE.
*******************************************************************************}

BEGIN
XERROR := 127 - (XCMAX + 8);
YERROR := 127 - (YCMAX + 8);

CLRSCR;
WRITELN; WRITELN;
WRITELN('----------------------------------------');
WRITELN('CORRELATION RESULT TABLE');
WRITELN('CORRELATION PEAK : ', CMAX, ' (MAXIMUM : 255)');
WRITELN('PEAK TO SIDELOBE RATIO : ', PSR:2:4, '');
WRITELN('X OFFSET : ', XERROR:3, '');
WRITELN('Y OFFSET : ', YERROR:3, '');
WRITELN('PRESS RETURN TO DISPLAY CORRELATION VALUES ');

READLN(INP3);
I := (XCMAX - 4);
WHILE I < (XCMAX + 5) DO
BEGIN
  J := (YCMAX - 4);
  WHILE J < (YCMAX + 5) DO
  BEGIN
    WRITE(PIX1[I,J]:4);
    J := J + 1;
  END;
  I := I + 1;
END;
PROCEDURE BCORR;

{**************************************************************************
** THIS PROCEDURE DOES THE CORRELATION **
** BETWEEN THE TWO IMAGES STORED IN THE **
** ARRAYS PIX1 AND PIX2. THESE TWO IMAGES **
** MUST BE STORED IN BILEVEL FORM, 0 OR 1(255). **
** THE OUTPUT WILL BE AN ERROR EXPRESSED IN **
** X PIXELS BY Y LINES. THIS WILL BE THE **
** DIFFERENCE OF THE COORDINATES OF THE **
** PEAK FROM THE CENTER OF THE IMAGE. **
** THE WINDOW (REFERENCE) DOING THE **
** CORRELATION WILL BE THE CENTER 16 X 16 **
** PIXELS OF IMAGE PIX1. **
**************************************************************************}

BEGIN
LOADREF;
CMAX := 0;
C := 91;
WHILE C < 148 DO
BEGIN
  D := 91;
  WHILE D < 148 DO
  BEGIN
    PIX1[C,D] := 0;
    A := 0;
    WHILE A < 16 DO
    BEGIN
      B := 0;
      WHILE B < 16 DO
      BEGIN
        IF (WINDOW[A,B] = PIX2[A + C,B + D]) THEN
        IF (PIX1[C,D] = 255) THEN WRITELN ELSE
            PIX1[C,D] := PIX1[C,D] + 1;
        B := B + 1;
      END;
      A := A + 1;
    END;
    IF (PIX1[C,D] > CMAX) THEN
BEGIN
CMAX := PIX1[C,D];
XCMAX := C;
YCMAX := D;
END;

D := D + 1;
END;
C := C + 1;
END;

SAV := (PIX1[XCMAX,YCMAX + 1] + PIX1[XCMAX,YCMAX - 1] +
       PIX1[XCMAX + 1,YCMAX] + PIX1[XCMAX - 1,YCMAX]) / 4;
PSR := PIX1[XCMAX,YCMAX] / SAV;
OUTERROR;
END;

PROCEDURE TCORR;

{************************************************************** **
** THIS PROCEDURE DOES THE SAME CORRELATION OPERATION AS BCORR **
** WITH THE EXCEPTION THAT IT OPERATES ON DATA ARRAYS WITH **
** FOUR POSSIBLE VALUES 0,1,2, OR 3. THE CORRELATION **
** SUM ARRAY IS ASSIGNED VALUES ACCORDING TO THE FOLLOWING **
** ALGORITHM: **
** (WINDOW - IMAGE) SUM **
** ----------------- ----- **
** 0 1 ** **
** 1 0 ** **
** 2 0 ** **
** 3 0 **
**************************************************************}

BEGIN
LOADREF;
WRITELN;
WRITELN(' PERFORMING THE TWO BIT CORRELATION ALGORITHM ');
WRITELN;
CMAX := 0;
C := 91;
CT1 := 0;
CT2 := 0;
WHILE C < 148 DO
  BEGIN
    D := 91;
    WHILE D < 148 DO
      BEGIN

PIX1[C,D] := 0;
A := 0;
WHILE A < 16 DO
BEGIN
B := 0;
WHILE B < 16 DO
BEGIN
CDIFF := WINDOW[A,B] - PIX2[A + C,B + D];
IF (CDIFF < 0) THEN CDIFF := (0 - CDIFF);
CASE CDIFF OF
0 : IF (PIX1[C,D] = 255) THEN WRITELN
ELSE PIX1[C,D] := PIX1[C,D] + 1;
85 : BEGIN
CT1 := CT1 + 1;
IF CT1 = 20 THEN
BEGIN
IF (PIX1[C,D] = 255) THEN WRITELN
ELSE PIX1[C,D] := PIX1[C,D] + 0;
CT1 := 0;
END;
END;
170 : BEGIN
CT2 := CT2 + 1;
IF CT2 = 20 THEN
BEGIN
IF (PIX1[C,D] = 255) THEN WRITELN ELSE
PIX1[C,D] := PIX1[C,D] + 0;
CT2 := 0;
END;
END;
END;
B := B + 1;
END;
A := A + 1;
END;
CT1 := 0; CT2 := 0;
IF (PIX1[C,D]) > CMAX THEN
BEGIN
CMAX := PIX1[C,D];
XCMAX := C;
YCMAX := D;
END;
D := D + 1;
END;
C := C + 1;
END;
SAV := (PIX1[XCMAX,YCMAX + 1] + PIX1[XCMAX,YCMAX - 1] +
        PIX1[XCMAX + 1,YCMAX] + PIX1[XCMAX - 1,YCMAX]) / 4;
PSR := PIX1[XCMAX,YCMAX] / SAV;
OUTERROR;
END;

PROCEDURE OUTSTATS;
{******************************************************
** THIS PROCEDURE OUTPUTS THE STATISTICS CALCULATED **
** IN THE FEATURE MATCHING ALGORITHM. **
******************************************************}
BEGIN
IF (XN11 = 0) OR (XN12 = 0) THEN WRITELN(' NO MAX VALUED
PIXELS IN CENTER FIELD OF VIEW ')
ELSE
BEGIN
XC11 := (XSUM11 / XN11);
YC11 := (YSUM11 / XN11);
XC12 := (XSUM12 / XN12);
YC12 := (YSUM12 / XN12);
XD := (XC11 - XC12);
YD := (YC11 - YC12);
CLRSCR;
WRITELN;
WRITELN('--------------------------------------');
WRITELN('FEATURE MATCHING ALGORITHM OUTPUT');
WRITELN('--------------------------------------');
WRITELN('IMAGE PLANE 11 STATISTICS');
WRITELN('NUMBER OF MAX VALUED PIXELS:','XN11:3:0','|');
WRITELN('XCENTROID: ','XC11:3:0','|');
WRITELN('YCENTROID: ','YC11:3:0','|');
WRITELN('--------------------------------------');
WRITELN('IMAGE PLANE 12 STATISTICS');
WRITELN('NUMBER OF MAX VALUED PIXELS:','XN12:3:0','|');
WRITELN('XCENTROID: ','XC12:3:0','|');
PROCEDURE FMATCH;

{****************************************************** **
** THIS PROCEDURE SCANS ARRAYS PIX1 AND PIX2 FOR **
** THE PEAK VALUES IN THE PICTURE. THE X AND Y **
** CENTROIDS ARE THEN CALCULATED FOR EACH PIC **
** AND AN OFFSET IS CALCULATED BASED ON THE **
** DIFFERENCE IN THE CENTROIDS. **
******************************************************}

BEGIN

WRITELN(' PERFORMING CENTROID CALCULATIONS '); WRITELN;

XSUM11 := 0.0; XSUM12 := 0.0;
YSUM11 := 0.0; YSUM12 := 0.0;
XN11 := 0; XN12 := 0;
FOR I := 32 TO 223 DO
  FOR J := 32 TO 223 DO
    IF (PIX1[I,J] = 255) THEN
      BEGIN
        XN11 := XN11 + 1;
        XSUM11 := XSUM11 + I;
        YSUM11 := YSUM11 + J;
      END;

  I := 32;
  WHILE I < 224 DO
    BEGIN
      J := 32;
      WHILE J < 224 DO
        BEGIN
          IF (PIX2[I,J] = 255) THEN
            BEGIN
              XN12 := XN12 + 1;
              XSUM12 := XSUM12 + I;
              YSUM12 := YSUM12 + J;
            END;
        END;
    END;
END;
```
J := J + 1;
END;
I := I + 1;
END;

OUTSTATS;
END;

PROCEDURE OUTOFFS;
BEGIN
CLRSCR;
WRITELN;
WRITELN('-----------------------------------------------');
WRITELN('HOT SPOT MATCHING RESULTS');
WRITELN('PLANE 1,1');
WRITELN('MAX VALUE: ',MAXSUM1:6:0,'');
WRITELN('LOCATION: ',YMAX1:3,',',XMAX1:3,'');
WRITELN('PLANE 1,2');
WRITELN('MAX VALUE: ',MAXSUM2:6:0,'');
WRITELN('LOCATION: ',YMAX2:3,',',XMAX2:3,'');
WRITELN('OFFSET: ',MDIFFY:3,',',MDIFFX:3,'');
WRITELN('-----------------------------------------------');
END;

PROCEDURE MAXMATCH;
{******************************************************************************************
** THIS SCENE MATCHING ALGORITHM ATTEMPTS TO MATCH THE HOT SPOT LOCATIONS OF THE MAX **
** 9 X 9 WINDOW IN EITHER PLANE PIX1 OR PIX2. **
******************************************************************************************}
VAR
TSUM,TI,TJ,TTI,TTJ : INTEGER;
```
BEGIN
CLRSCR;
WRITELN;
WRITELN('PERFORMING HOT SPOT MATCHING ALGORITHM');
WRITELN;
I := 32;
MAXSUM1 := 0.0;
WHILE (I < 224) AND (MAXSUM1 <> 20655) DO
BEGIN
  J := 32;
  WHILE J < 224 DO
  BEGIN
    TSUM := 0;
    FOR TI := -4 TO 4 DO FOR TJ := -4 TO 4 DO
      BEGIN
        TTI := I + TI;
        TTJ := J + TJ;
        TSUM := TSUM + PIX1[TTI, TTJ];
      END;
    IF (TSUM > MAXSUM1) THEN
      BEGIN
        XMAX1 := I;
        YMAX1 := J;
        MAXSUM1 := TSUM;
      END;
    J := J + 1;
  END;
  I := I + 1;
END;
MAXSUM2 := 0.0;
I := 32;
WHILE (I < 224) AND (MAXSUM2 <> 20655) DO
BEGIN
  J := 32;
  WHILE J < 224 DO
  BEGIN
    TSUM := 0;
    FOR TI := -4 TO 4 DO FOR TJ := -4 TO 4 DO
      BEGIN
        TTI := I + TI;
        TTJ := J + TJ;
        TSUM := TSUM + PIX2[TTI, TTJ];
      END;
    IF (TSUM > MAXSUM2) THEN
      BEGIN
        XMAX2 := I;
      END;
  END;
END;
YMAX2 := J;
MAXSUM2 := TSUM;
END;

J := J + 1;
END;

I := I + 1;
END;

MDIFFX := XMAX1 - XMAX2;
MDIFFY := YMAX1 - YMAX2;
OUTOFFS;
END;

PROCEDURE OFFS;
BEGIN
CLRSCR;
WRITELN;
WRITELN('---------------------------------------------------------------------');
WRITELN(' HOT SPOT MATCHING RESULTS');
WRITELN('---------------------------------------------------------------------');
WRITELN(' PLANE 1,1');
WRITELN(' MAX VALUE : ','MXS1:3,'');
WRITELN(' LOCATION : ','YM1:3','','XM1:3','');
WRITELN(' PLANE 1,2');
WRITELN(' MAX VALUE : ','MXS2:3,'');
WRITELN(' LOCATION : ','YM2:3','','XM2:3','');
WRITELN(' OFFSET : ','DIFY:3','','DIFX:3','');
WRITELN('---------------------------------------------------------------------');
END;
PROCEDURE MAXONLY;

{************************************************************************** *
** THIS PROCEDURE USES ONLY THOSE VALUES WHICH HAVE **
** BEEN ASSIGNED THE MAX VALUE (255) BY THE **
** SEGMENTATION ALGORITHM. **
**************************************************************************}

VAR
TS,TX,TY,TTX,TTY : INTEGER;

BEGIN
CLRSCR;
WRITELN;
WRITELN(' PERFORMING MAXONLY KERNEL SUM MATCHING ON
PLANES 11 AND 12.');
WRITELN;
I := 32;
MXS1 := 0;
WHILE (I < 224) DO
BEGIN
  J := 32;
  WHILE J < 224 DO
  BEGIN
    TS := 0;
    FOR TX := -4 TO 4 DO FOR TY := -4 TO 4 DO
    BEGIN
      TTX := I + TX;
      TTY := J + TY;
      IF PIXL[TTX,TTY] = 255 THEN TS := TS + 1;
    END;
    IF (TS > MXS1) THEN
    BEGIN
      WRITELN(' NEW PEAK AT',I:3,',',J:3);
      READLN(INP2);
      XM1 := I;
      YM1 := J;
      MXS1 := TS;
    END;
    J := J + 1;
  END;
  I := I + 1;
END;
MXS2 := 0;
I := 32;
WHILE (I < 224) DO
BEGIN
  J := 32;
WHILE J < 224 DO
BEGIN
  TS := 0;
  FOR TX := -4 TO 4 DO FOR TY := -4 TO 4 DO
    BEGIN
      TTX := I + TX;
      TTY := J + TY;
      IF PIX2[TTX,TTY] = 255 THEN TS := TS + 1;
    END;
  IF (TS > MXS2) THEN
    BEGIN
      XM2 := I;
      YM2 := J;
      MXS2 := TS;
    END;
  J := J + 1;
END;
I := I + 1;
END;
DIFX := XM2 - XM1;
DIFY := YM2 - YM1;
OFFS;
END;

{****************************************************** ** MAIN PROGRAM **
******************************************************}
BEGIN
SCRSET;
CASE INP3 OF
  '1' : GTHRESH;
  '2' : GTWOBIT;
  '3' : ATHRESH;
  '4' : ATWOBIT;
  '5' : BCORR;
  '6' : TCORR;
  '7' : FMATCH;
  '8' : MAXMATCH;
  '9' : MAXONLY;
END;
END.
APPENDIX B

IMAGE PROCESSING SYSTEM
This appendix briefly describes the image processing system used to perform the algorithm simulations. The system is a personal computer based image processing station, capable of performing a variety of processing tasks. The plug-in board is manufactured by Weiland Systems Design, El Paso Texas. The PCIP 100 Single Board Monochrome System comes complete with a Vidicon Television Camera and a complete software library of image processing routines, system initialization and checkout programs, and interfaces to common programming languages such as Borland Turbo Pascal, Microsoft Fortran 77, Microsoft C, and Microsoft Pascal. Figure 16 illustrates the system setup.

Figure 16. Image Processing System Configuration.
The host personal computer is an IBM AT compatible, PC's Limited 286. The Weiland Board simply plugs into an expansion slot and comes equipped with the necessary connections and cables to interface to the imaging monitor and the vidicon camera. The board contains 512K bytes of memory, 128K of which are addressable from PC address space. This is enough memory for eight 256 x 256 x 8 bit images, two of which are available to the PC. Through software utilities supplied with the board, the images are switched into and out of the available memory space. The board outputs RS170 standard video to the imaging monitor, a Magnavox RGB Monitor 40.
BIBLIOGRAPHY


