1

# HYBRID OPTICAL/DIGITAL ARCHITECTURE FOR DISTORTION INVARIANT PATTERN RECOGNITION

THESIS

John D. Cline
Captain, USAF

AFIT/GEO/ENG/89D-2

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GEO/ENG/89D-2

1

# HYBRID OPTICAL/DIGITAL ARCHITECTURE FOR DISTORTION INVARIANT PATTERN RECOGNITION

THESIS

John D. Cline
Captain, USAF

AFIT/GEO/ENG/89D-2

Approved for public release; distribution unlimited

AFIT/GEO/ENG/89D-2

# HYBRID OPTICAL/DIGITAL ARCHITECTURE FOR DISTORTION INVARIANT PATTERN RECOGNITION

THESIS

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

John D. Cline, B.B.A., B. S. E. E.
Captain, USAF

December 1989

# ACKNOWLEDGEMENTS

Throughout this thesis effort, I received technical assistance and encouragement from many people. Frequently these personal contacts provided the necessary tools to overcome seemingly insurmountable barriers. Although my appreciation goes out to a great number of people I've met while here at AFIT, a few will always be remembered as the guys that got me through it all.

Everything I know about Fourier optics was taught to me by a man named Dr. Steve Rogers (Major), and for that, as well as his technical leadership, I am deeply indebted. I also wish to thank my thesis committee members Dr. Matthew Kabrisky, and Dr. James Mills (Lt Col), for their kind and wise words as the thesis developed.

Thanks also go out to my friends, Captains John Murphy and Tim Childress, for their insights into camera problems and CGH alignment respectively.

The engineers at Perkin-Elmer, James Logue and David Zweig, graciously loaned us their fantastic CGH, complete with technical expertise only a phone call away. For that, I am extremely grateful.

Osvaldo Perez is the last, but certainly not the least, of the engineers I'd like to thank. His help in avoiding the joint transform correlator pitfalls, bench alignment, and simulations were critical to the few successes achieved during this thesis effort.

Finally, I'd like to thank my wonderful wife Terry for her support and patience, despite the many long evenings and countless weekends, that left her, and trusty dog Smokey, home alone.

John D. Cline

## Table of Contents

Page

## List of Figures

Figure Page

## ABSTRACT

This research investigated optical techniques for pattern recognition. An optical joint transform correlator was implemented using a magneto-optic spatial light modulator, and a charge coupled device (CCD) camera and framegrabber under personal computer (PC) control. A hybrid optical/digital architecture that could potentially perform position, scale, and rotation invariant pattern recognition using a computer generated hologram (CGH) was also implemented.

The joint transform correlator was tested using forward looking infrared (FLIR) imagery containing tactical targets, and gave very good results. New techniques for binarizing the FLIR inputs and the fringe pattern of the joint transform were discovered. The input binarization used both scene average and a localized energy normalization technique for binarization. This resulted in reduced scene background, while retaining target detail. The fringe binarization technique subtracted the Fourier transform of the scene from the joint transform, and binarized on the average difference. This new technique was a significant improvement over recent published designs.

# HYBRID OPTICAL/DIGITAL ARCHITECTURE FOR DISTORTION INVARIANT PATTERN RECOGNITION

## I. INTRODUCTION

Machine intelligence for pattern recognition has long been a dream of both industrial and military engineers. Imagine a robot on a flight line capable of autonomously navigating its way between and through various maintenance tasks. How about a smart satellite that could not only scan the earth's surface for enemy movement, but could also alert one of potential aggressive acts? Perhaps we could even develop a tactical fighter missile that locates and destroys its target based on the target's shape (as opposed to its temperature like the easily confused heat seeking missile). A pattern recognition system that works well in a myriad of uncontrolled environmental conditions is the essential missing component for these and many other intelligent systems.

To be effective for these applications, a pattern recognition system must work regardless of the target's illumination, position, scale and orientation, or its partial obstruction by scene clutter. It must provide results in near real time, with a high probability of target detection as well as a low probability of false alarm.

Over the years, much has been written about the diverse field of pattern recognition. To date, this work has yielded techniques thac solve onl} the simplest "toy problems." The majority of techniques showing any promise employ sophisticated computer algorithms requiring large computers and long calculation times. Unlike the many pattern recognition techniques employing complex computer algorithms, optical information processing does not require large scale computers, and can potentially give results at the speed of light.

## 1.1 Background

In his classic text Introduction to Fourier Optics, Joseph W. Goodman discusses a number of processes that are the usual points of reference for all discussion concerning optical information processing. He discusses the Fourier transforming property of lenses [1:83-90], the Vander Lugt filter [1:171-177], and the joint transform correlator [1:194]. He also suggests that the well known matched filter correlator is extremely sensitive to changes of scale or rotation of scene objects with respect to a reference template [1:183-184].

Employing coordinate transformations, David Casasent and Demetri Psaltis of Carnegie Mellon University discovered that they could improve the probability of detection of rotated and scaled objects. They used a computer generated hologram (CGH) to transform light intensity distributions from x-y coordinates to log-polar coordinates (logarithm of the radial coordinate). Using these features they were able to employ matched filter correlation that was both rotation and scale invariant [2:77-84]. However, even though they could decide whether or not a rotated and scaled object was in a particular scene, they could not locate it.

Under the direction of Major Steven Rogers, Dr. Matthew Kabrisky, and Lieutenant Colonel James Mills, thesis students at the Air Force Institute of Technology (AFIT) have continued to pursue pattern recognition techniques that are both scale and rotation, as well as position invariant [3]. Horev proposed a technique [4] implemented as a computer algorithm by Kobel and Martin [5] that provided all of these features. Schematically pictured in Figure 1, this technique first performed the two dimensional Fourier transform (FT) on the input object, then performed the log-polar coordinate transformation (LPCT) on the magnitude of the Fourier transform ($|FT|^2$), followed by correlation with a template. Casasent and Psaltis proposed an optical implementation of these few steps, but the Kobel-Martin-Horev (KMH) algorithm didn't stop there. Using the correlation data, they found the proper scale and rotation of the target within the scene, and rectified (corrected) the template to reflect these. This, coupled with some additional processing, allowed them to locate the target.

Troxel augmented the KMH algorithm with artificial neural network processing to perform target recognition on range imagery [6]. Using features that described the shape

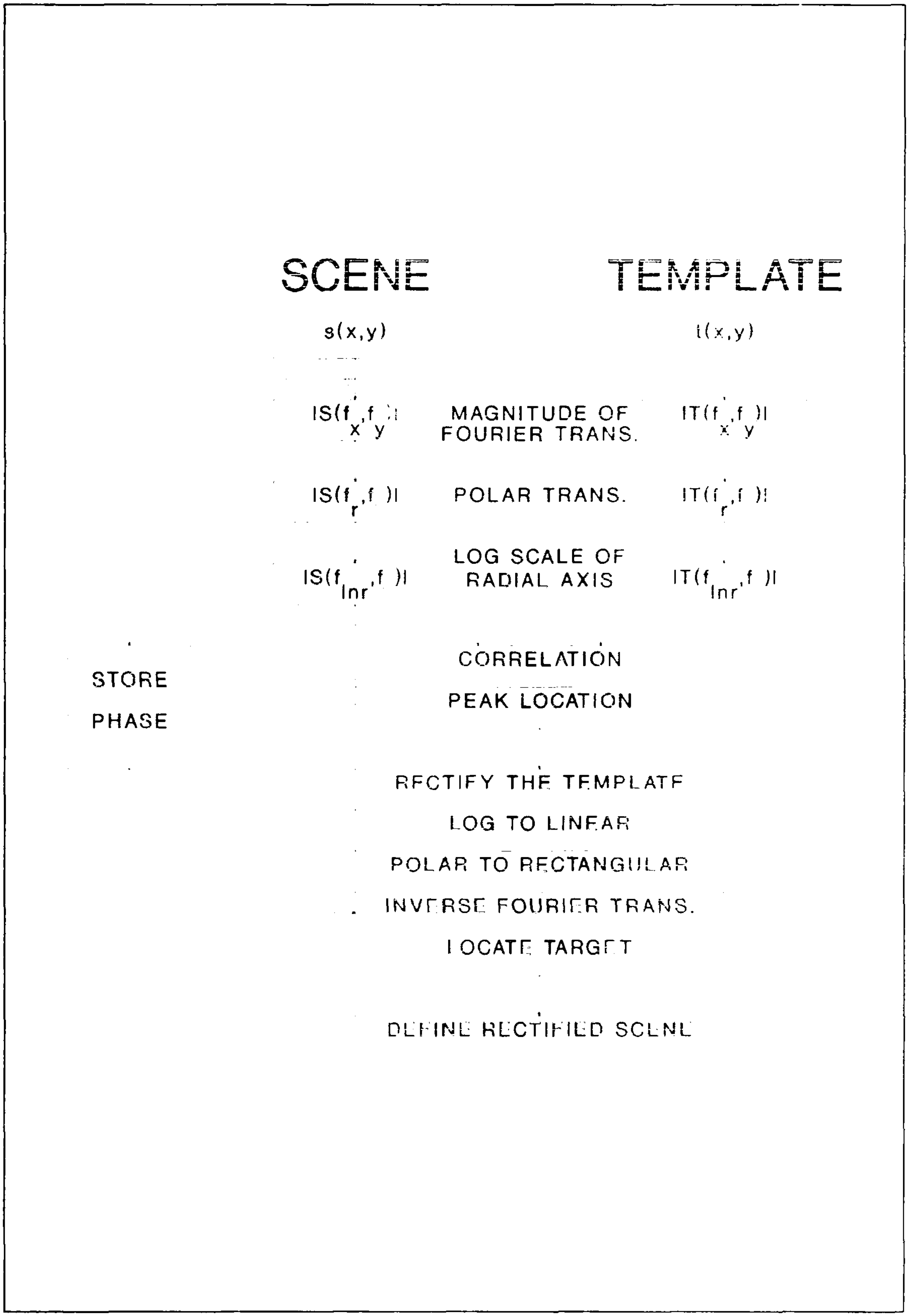

**Figure 1.** Horev Algorithm

of autocorrelations and crosscorrelations of target objects, his multilayer perceptron was trained to classify various targets. However, like Kobel, Martin and Horev's work, his algorithm was far too slow to be considered much more than an interesting result.

The need to reduce the computational load on this pattern recognition technique was paramount if it was to have any application. The obvious choice at AFIT was to attempt an optical implementation of the KMH algorithm. One optical architecture was originally suggested by Miazza [7]. The greatest hardware requirements for this architecture were two types of spatial light modulators (SLMs), a magneto-optic SLM (MOSLM) and a liquid crystal light valve. These devices were still in developmental stages, so they were very expensive, in fact, AFIT has yet to receive adequate financial support to purchase the liquid crystal light valve. However, that has not halted research here.

Using the Casasent-Psaltis technique, Mayo implemented the LPCT of various objects. He began his work using a Litton MOSLM as an input image device, but had to return to glass slides when the device failed [8]. This first attempt set the foundation for further developments by Childress and Walrond [9]. Their work aiffered from Mayo's in that they performed the LPCT of the $|FT|^2$ of input images, not just the LPCT of the images themselves. Moreover, they verified that correlation in this LPCT-$|FT|^2$ space could provide the proper scale and rotation for template rectification.

However, their research stopped short of using this scale and rotation information to continue target identification in the original scenes. Their work suggested the viability of optical implementation, but they had no continuous process (the steps were implemented piecemeal, and a mainframe computer, as well as human judgement, were required for intermediate calculations), so they fell short of a real time, continuous process.

The joint transform correlator (JTC) suggested by Weaver and Goodman [10], has gained new interest in professional publications recently. Javidi, Gregory, and Horner [11] have done theoretical and experimental research with both liquid crystal televisions (LCTVs) and MOSLMs. Interest has peaked because these two devices can make possible real time implementation of a correlator. However, these correlators are not scale and rotation invariant.

## 1.2 Problem Definition

A large step in optically implementing a KMH type pattern recognition scheme as a continuous process is the development of a working optical correlator. This correlator could be used to find the proper scale and rotation as suggested by the KMH algorithm. It could then be used to correlate the original scene (or perhaps an enhanced scene) with a rectified template to acquire the target's location in the scene. Once the target was acquired the correlator could be used as a tracker.

### 1.3 Scope of Thesis

The first major goal of this thesis was to develop software for personal computer (PC) control of the MOSLM and the two charge coupled device (CCD) cameras connected to the framegrabber card. Since development of techniques applicable to real data was the goal, this process also included preparation of available forward looking infrared (FLIR) image data as sample input images for display on the MOSLM. These FLIR files were eight bit grey scale arrays of 240X640 pixels, so considerable effort needed to be taken to make them useable in a JTC that employed a binary 128X128 pixel MOSLM. The binarization of images for use in hybrid optical systems is a major problem to be addressed, because the techniques used for binarization can drastically affect results.

The next goal was to implement in hardware an optical JTC like those suggested in the publications. Using FLIR imagery, as opposed to some sort of "toy" data, provided a significant first step toward evaluating this technology for practical applications. Several improvements were made upon published designs of the JTC, and these data were used in experiment, as well as simulation, for verification.

The final goal was to design and test a hybrid optical/digital architecture for distortion invariant pattern recognition. The system, pictured in Figure 2, could potentially perform the LPCT process necessary for position scale and rotation invariant (PSRI) pattern recognition. As part of this third goal, a sample of the CGH created by Mayo [8] and Hill [12] was compared with one temporarily on loan from Perkin-Elmer. Correlation of test objects using the LPCT-$|FT|^2$ features was also performed.

## 1.4 Approach

Optical data flow through this architecture followed one of two paths (see Figure 2). First binarized images were displayed on the MOSLM by the PC. Next, the $|FT|^2$ of the image was detected at CCD1, and the LPCT of the image was detected at CCD2. By throwing a manual switch, the operator could decide which CCD camera would be connected to the framegrabber in the PC. (This manual switch could easily be replaced with a PC controllable one in the future.) Once the PC had captured the data, it could perform calculations on it, and where appropriate, display binarized versions of these captured frames on the MOSLM.

To acquire the LPCT-$|FT|^2$ features, a binarized image was displayed on the MOSLM. CCD1 detected the $|FT|^2$ of the image and the PC displayed its binary representation on the MOSLM. CCD2 detected the LPCT of this binarized $|FT|^2$ and the PC saved the frame for later use.

To perform correlation in this architecture the PC displayed a frame on the MOSLM. This frame had a template ($t(x,y)$) centered on the bottom half and a scene ($s(x,y)$) centered on the top half. CCD1 detected the joint power

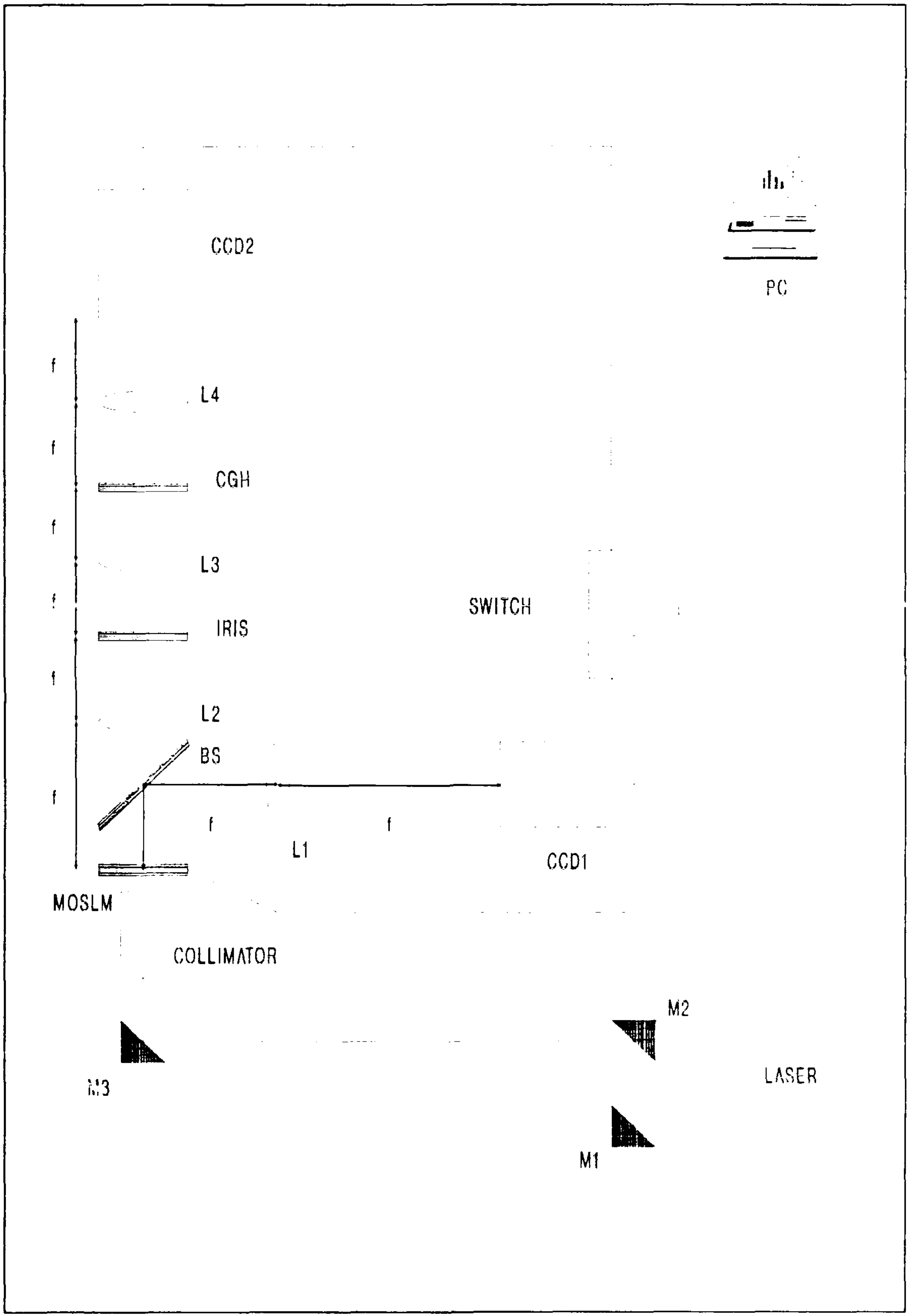

**Figure 2.** Hybrid Optical/Digital Architecture

spectrum, or joint transform, of t and s. The PC binarized this joint transform and displayed it on the MOSLM. CCD1 then detected what will be called the correlation plane. On axis was the autocorrelation of t ($C_t$) plus the autocorrelation of s ($C_s$). Above and below these autocorrelations was the crosscorrelation of t and s ($C_{ts}$). The PC located the correlation peaks in the crosscorrelations, interpreted the results as candidate target locations, (or perhaps scales and in-plane rotations), and reported results.

Using the LPCT-$|FT|^2$ features of a scene and a template, correlation was to provide the proper scale and in-plane rotation of the target in the scene. This was true since in this feature space shifts of the correlation peak along one axis relate to scale changes, while shifts along the other axis relate to in-plane rotations. Using this information, the template could be rectified to reflect these, and correlation in the original scene space would yield the target's location.

## 1.5 Overview of Thesis

This thesis is divided into six chapters and seven appendices. The chapters are used to discuss general concepts and results, while the appendices are reserved for details on equipment as well as computer code.

Chapter I, this chapter, serves as an introduction to the thesis topic. Chapter II discusses basic concepts such as correlation, joint transform and coordinate transformation, and describes in greater detail, the hybrid architecture. Chapter III describes the JTC, discusses some important engineering decisions associated with it, and presents experimental results. Chapter IV describes the LPCT and presents experimental results for both the Perkin-Elmer sample CGH as well as the Mayo-Hill CGH. Chapter V describes a few details concerning the PSRI architecture and presents experimental results of the correlation of the LPCT-$|FT|^2$ features. Finally, chapter VI summarizes the results, presents conclusions, and provides recommendations for potential future research effort.

Appendices A through C provide detailed information concerning the MOSLM, framegrabber system, and sample FLIR data respectively. Appendices D through F discuss the use of tools available on various systems that may assist in documentation. Finally, Appendix G describes software used for equipment control for the continuous process that makes up the hybrid system architecture.

# II. BASIC CONCEPTS

This chapter discusses several concepts central to this thesis. First, theoretical developments of correlation and the joint transform correlator will be presented. Next, the concept of a scale and rotation invariant feature space will be presented, followed by discussion of the KMH algorithm. Finally, the hybrid optical/digital implementation of a variation on the KMH algorithm will be described.

## 2.1 Correlation

Correlation is a mathematical process that provides an estimate of the similarity between two things. The process is simply described in layman's terms for target recognition. A picture of what the target looks like (known as a template) is compared to all locations on a picture of a much larger scene (imagine a transparent template being passed over the scene picture). At each location a number (the correlation) is returned that represents the similarity between the template and the scene at that location. If the scene is appropriately normalized, then at the locations where the scene and template agree most, the correlation will be the largest (this will be referred to as the correlation peak), and, if the correlation peak is greater than some threshold, that location could be identified as a target. The shape of the correlation peak has also been used for locating targets and is a possible alternative or supplement to this threshold detection technique [6].

Distinction between autocorrelation, crosscorrelation, and just plain correlation is necessary at this point. Autocorrelation refers to the correlation of an object with itself. Crosscorrelation refers to correlation of one object with another. In most articles, use of the word correlation without a prefix will mean crosscorrelation. Another point of confusion to look out for is the use of the word autocorrelation for a crosscorrelation that correctly identifies a target. These will be avoided from here on in this thesis.

Goodman provides this mathematical description [1:175]:

$$C_{st}(x,y) = s(\alpha,\beta) \star t(\alpha,\beta) \tag{1}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(\alpha,\beta)\; t^{*}(\alpha-x,\beta-y)\; d\alpha d\beta \tag{2}$$

$$= \Im^{-1}\{S(f_x,f_y)T^{*}(f_x,f_y)\} \tag{3}$$

where $\alpha$ and $\beta$ are dummy variables, s and t are scene and template input image intensity distributions, and S and T their respective Fourier transforms. The symbol $\Im^{-1}$ represents the inverse Fourier transform, $\star$ stands for correlation, and $^{*}$ means complex conjugate. The variables $f_x$ and $f_y$ are used to represent the spatial frequencies associated with the x and y cardinal directions. Note that $C_{st}$ represents the crosscorrelation of s and t. By replacing t with s, the above equations would represent the autocorrelation of the scene. However, as was mentioned in chapter 1, the notation will be $C_s$, not $C_{ss}$. Similarly, $C_t$ will denote the autocorrelation of the template.

## 2.2 Joint Transform Correlation Theory

One way of acquiring the $C_{st}$ of real valued inputs was suggested by Weaver and Goodman [10]. Figure 3 pictures a scene and a template, both displayed in a single input plane. The scene is centered in the top half of the plane at a location (0,Y/2) and the template at (0,-Y/2). The converging lens, placed exactly one focal length away from the input and output planes provides a Fourier transformation on the input. That is to say that if the input plane is a two dimensional transmission function, and it is illuminated with a uniform amplitude plane wave, then light appearing at the output plane would have a distribution equal to the Fourier transform of the input plane's transmissivity both in phase and amplitude. The spatial frequencies $f_x$ and $f_y$ can be related to physical dimensions by:

$$f_x = x/\lambda f \quad \text{and} \quad f_y = y/\lambda f \qquad (4)$$

where $\lambda$ is the wavelength of light and f is the focal length. Furthermore, shifts of an object in the input plane results only in phase variations in the Fourier transform in the output plane. These are well known facts, and further development can be found in any book containing chapters on Fourier optics. Therefore, no further development will be presented on this concept here.

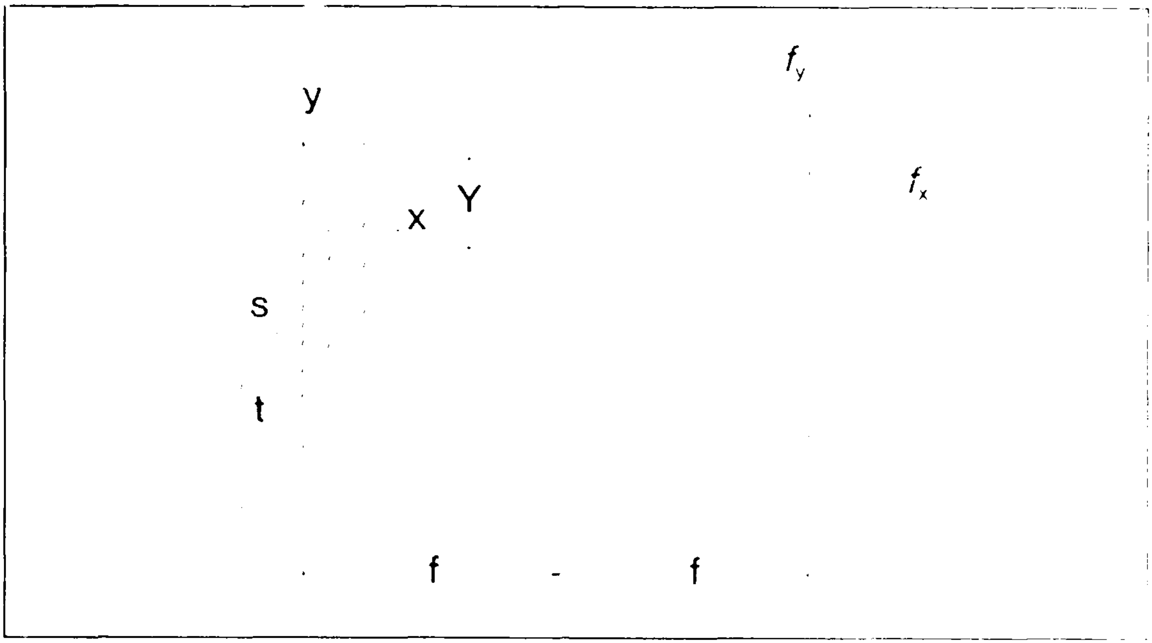

**Figure 3.** Goodman's Optical Arrangement

Given that this optical arrangement can Fourier transform the input, and that shifts in the input [illegible] phase differences in the output, Goodman's discovery can now be developed. The input plane can be represented by the equation:

$$i(x,y) = s(x, y-Y/2) + t(x, y+Y/2)$$

whose Fourier transform is:

$$I(f_x,f_y) = S(f_x,f_y)\, e^{-j\pi f_y Y} + T(f_x,f_y)\, e^{+j\pi f_y Y}$$

However, available detectors only measure the intensity of light incident upon them, not the amplitude and phase, [illegible] this intensity distribution can be represented as:

$$\begin{aligned} |I(f_x,f_y)|^2 &= \left|S(f_x,f_y)\, e^{-j\pi f_y Y} + T(f_x,f_y)\, e^{+j\pi f_y Y}\right|^2 \\ &= |S(f_x,f_y)|^2 + |T(f_x,f_y)|^2 \\ &\quad + S(f_x,f_y)T^*(f_x,f_y)\, e^{-j2\pi f_y Y} \\ &\quad + S^*(f_x,f_y)T(f_x,f_y)\, e^{+j2\pi f_y Y} \end{aligned}$$

If this intensity distribution is now placed in the input plane of this optical arrangement, the resultant output plane will be:

$$O(x,y) = C_s(x,y)+C_t(x,y)+C_{st}(x,y-Y)+C_{ts}(x,y+Y) \qquad (8)$$

Figure 4 depicts what this output plane will look like in general. Notice that the two crosscorrelations appear above and below the central autocorrelations and they are offset from the origin by the distance Y. It can be shown that to avoid overlap of the auto- and crosscorrelations, the input scene and template must be separated by at least the sum of 3/2 of the larger's vertical extent plus 1/2 of the smaller's vertical extent. This issue is important and is discussed in greater detail in Chapter III.

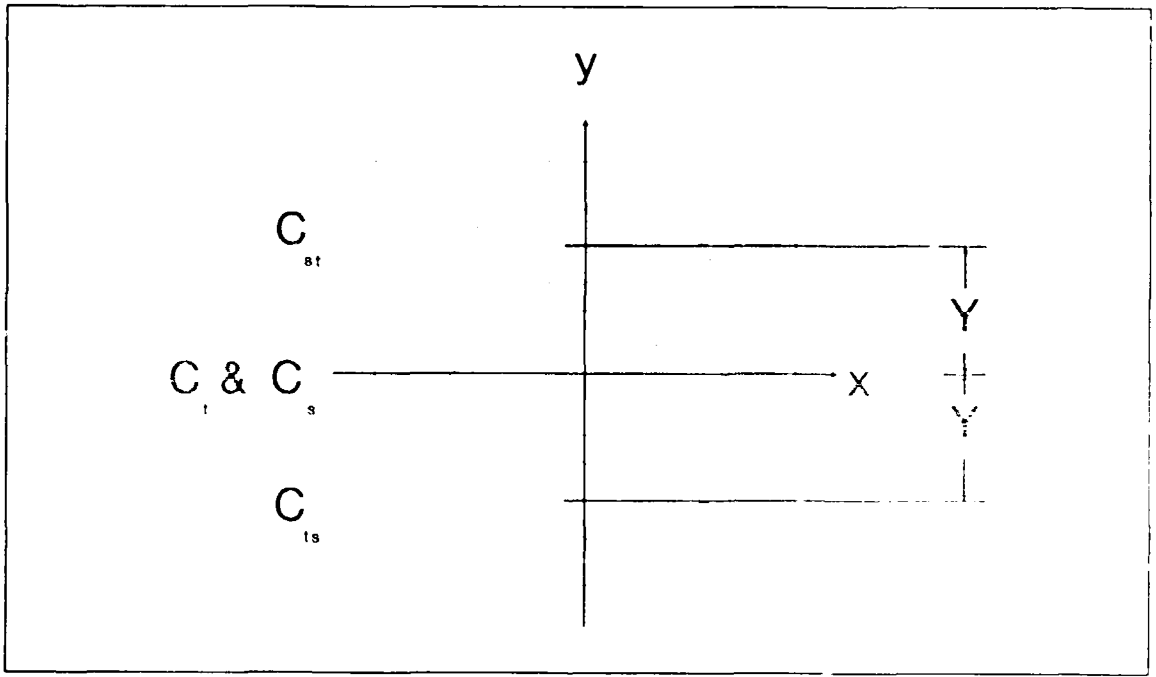

**Figure 4.** Typical Correlation Plane

## 2.3 Scale and Rotation Invariance

One weakness of crosscorrelation as a tool for locating targets in a scene is that frequently the target may not have the same scale or orientation as the template being used. How much impact this might have on crosscorrelation's ability to locate targets will vary greatly with the application. Clearly, a circle's shape doesn't vary with rotation. However, crosscorrelations of irregular shapes, especially edge enhanced ones, will suffer dramatically when the template is improperly scaled or rotated. In general, more than a few degrees of rotation, or a few percent of scale change will typically make a target unrecognizable by correlation techniques. One solution to this problem would be to crosscorrelate scenes with templates of all possible scales and orientations. This would work, but the large amount of processing necessary would make such a system too slow or large to be useful.

Cassasent and Psaltis proposed a feature space that would provide correlations that were scale and in-plane rotation invariant [2]. They suggested that the magnitude of the Fourier Transform be mapped in rectangular coordinates with $f_{\log(r)}$ along the vertical axis, and $f_{\theta}$ along the horizontal axis. In this feature space, crosscorrelation would turn changes in scale to vertical shifts and changes in-plane rotation to horizontal shifts of the correlation peak. The problem was that although a strong correlation peak with these features would indicate the presence of a target, it would not locate it within the scene (Not to mention that out-of-plane rotations would still require multiple templates).

### 2.4 KMH Algorithm

To tackle the in-plane rotation problem, Horev used these LPCT-$|FT|^2$ features to determine the scale and rotation of the target within the scene [4]. Using this information the template was rectified to reflect the proper scale and rotation. The amplitude of this new template's Fourier spectrum was combined with the phase of the scene, then inverse Fourier transformed. The result was a scene with an enhanced target. Kobel and Martin crosscorrelated this new scene with the rectified template to acquire the target's location [5].

### 2.5 Hybrid Optical/Digital Architecture

As was mentioned in Chapter I, attempts at implementing this algorithm in optics has occupied a number of people here at AFIT over the last few years, but so far, only limited results have been achieved. A hybrid optical/digital architecture was implemented for this thesis effort that employed the JTC discussed in section 2.2. It also employed a CGH to perform the necessary coordinate transformation. Two significant differences between this architecture and the KMH algorithm are that the input device is binary, and no capability for phase reinsertion is provided.

A schematic diagram of the architecture was presented as Figure 2 in Chapter I. Figure 5, on the following page is a flowchart of the processing performed by this architecture. Suggestions for improvement of this architecture are offered in Chapter VI.

## 2.6 Summary

This chapter discussed several concepts key to the understanding of this thesis. Correlation was defined and the principles of the JTC were presented. The notion of a scale and rotation invariant feature space and the KMH algorithm were also presented. Finally, the hybrid optical/digital implementation of a variation of the KMH algorithm was described.

Chapters III, IV, and V, discussing the JTC, LPCT, and the hybrid system architecture respectively, should now be easier to comprehend.

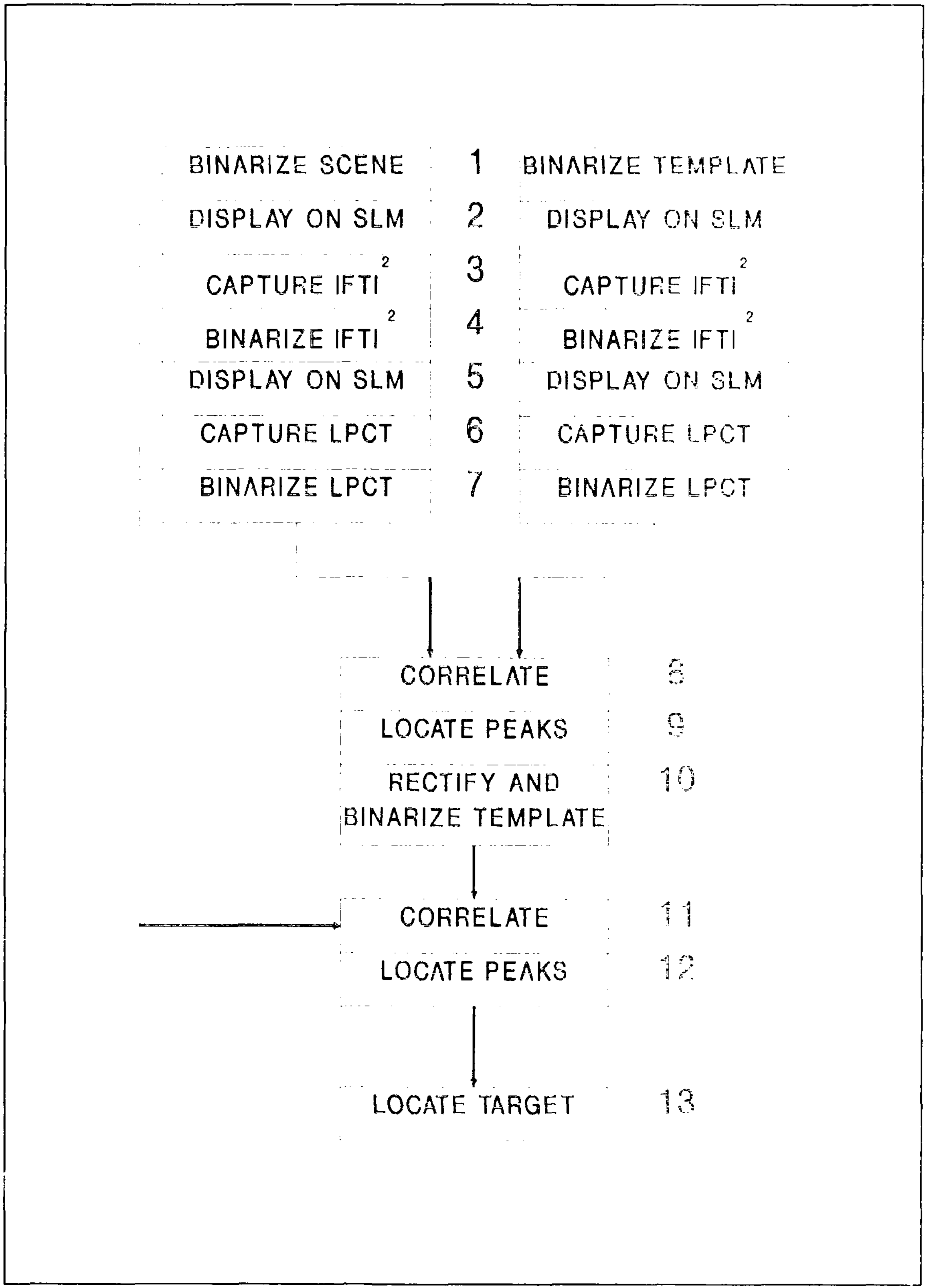

**Figure 5.** Hybrid optical/digital architecture flow diagram.

# III. JOINT TRANSFORM CORRELATOR (JTC)

This chapter presents information about the JTC. First, a summary of published material on the subject will be presented. Next, engineering issues relating to the implementation of the optical JTC will be presented. Finally, theoretical and experimental results using FLIR data will be presented. The design and implementation of a JTC is the most significant goal of this thesis effort.

## 3.1 Current Knowledge

In 1966, Joseph W. Goodman and C. S. Weaver published the article "A Technique for Optically Convolving Two Functions" in Applied Optics [10] (this work was sponsored by the USAF Avionics Laboratory). From their work using film, the experimental JTCs employing MOSLMs, LCTVs and deformable mirrors [13] have evolved.

More recently, a number of articles have been written by various authors on the JTC. The attraction of this technique of correlation is that: 1) correlation at television frame rates (30 frames/second) is possible; and that 2) alignment is not nearly as critical as the Vander Lugt filtering technique [14].

In an article by Bahram Javidi (who has received support from the Rome Air Development Center detachment operating at Hanscomb AFB, MA, Dr. Joseph Horner), the performance of various techniques of correlation are compared [15]. He concludes that a binary JTC (the Fourier plane intensity is binarized based upon a median value threshold) outperforms the Vander Lugt matched filter correlator, the binary phase only matched filter correlator, and the grey scale (or classical) JTC.

The intuitive reasoning here might be that the binarized fringe pattern would be closer to a pure sinusoidal pattern (actually it is more like a square wave), which would result in sharper peaks (more like an ideal delta function) in the correlation plane than that of the classical JTC. Vander Lugt matched filtering should yield results similar to the classical JTC, so the improvement there is plausible. The reason for the improvement over the binary phase only matched filter technique was not obvious, but was not investigated during this thesis effort.

Also notable amongst the researchers is Don A. Gregory (with U. S. Army Missle Command, Redstone Arsenal, Alabama), who has performed research using LCTVs and sponsored other work performed by Francis T. S. Yu employing MOSLMs. They co-authored an article "Illumination dependence of the joint transform correlation" that appeared in Applied Optics [16]. In this article, the problem of having a template that is or a different intensity than a target within the scene is presented. This problem has always received much attention here at AFIT, and the AFIT solution is to locally energy normalize the scene before trying to do any further processing [17].

In another article by Yu and Gregory, the ill effects of fringe binarization are addressed [18]. The assumption in this article was that fringes were binarized based upon some fixed threshold value like Javidi's work. However, that was not the case for the technique used in this thesis (see Figure 6).

Binarization of the fringes was performed by subtracting the intensity detected in the Fourier plane from displaying just the scene ($|S|^2$) from that detected from displaying both the scene and the template ($|S + T|^2$ or the joint transform). This difference was then compared to a threshold. If it was greater than the threshold, then that pixel would receive a one, if less, a zero.

This technique yielded three benefits: 1) it eliminated $C_s$ from the correlation plane which allowed the scene and template to be closer togetner (refer to the mathematicai development in section 2.2 if this is not clear); 2) it improved the quality of the fringes displayed on the MOSLM, as will be evidenced later in this chapter, and therefore increased the size of the crosscorrelation peaks; and 3) it eliminated many of the false alarms and missed detections that Yu and Gregory identified for their binarization technique.

## 3.2 Engineering Issues

This section presents information relating to engineering decisions made in implementing the JTC with real

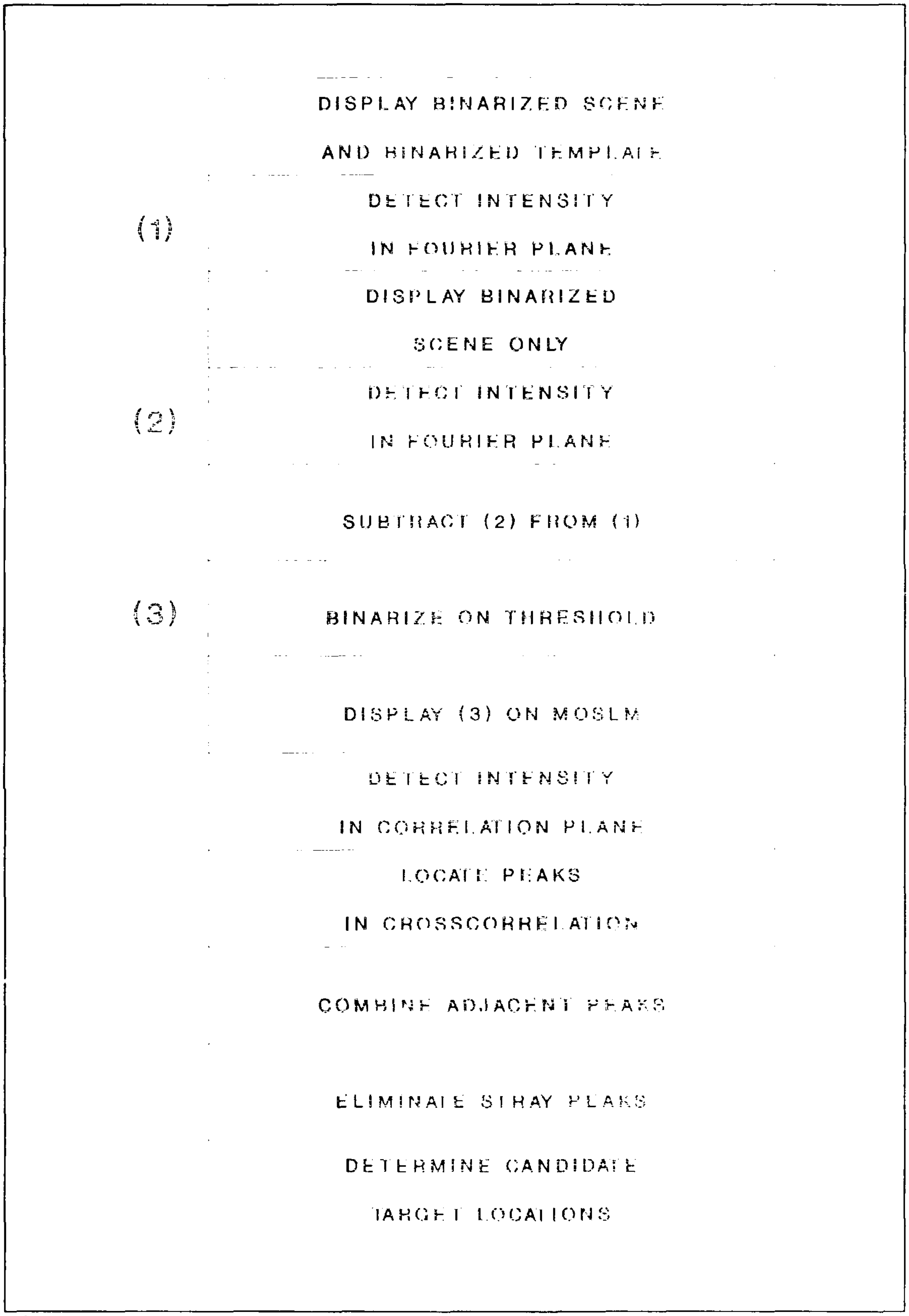

**Figure 6.** Joint Transform Correlator Flow Diagram

FLIR data. It is divided in six subsections: 1) Detection Limitations; 2) MOSLM Limitations; 3) Input Data; 4) Fringe Binarization; 5) Spatial Frequency Components; and 6) Correlation Plane Interpretation.

3.2.1 Detection Limitations. In Goodman's original setup using film, he was concerned about the ability of the film to maintain its square law operation over a broad dynamic range of intensities. This is also a problem for CCD cameras. In fact, even more significant is a phenomena known as "blooming" that was discussed by Mayo [8:37], and reitterated by Childress and Walrond [9:23]. When a region of the detector array is very brightly illuminated, pixels along the horizontal axis of the array will also be excited into saturation. For these reasons the selection of neutral density filters to be placed in front of the camera was critical. The decision was complicated further by the fact that the same camera was used to detect both the Fourier plane and the correlation plane, which typically had different energy distributions. A trial and error technique was employed for filter selection.

Several other sources of error occured in the detection process. Typically, detector arrays are made up of discrete pixel elements (the SONY model XC-38 camera used in this experiment was a 384(H) X 491(V) array). Response of the individual pixel elements is not identical, therefore the first source of error is introduced. Since the camera's output was NTSC video, an analog signal, some error must be occuring in the process. The AT&T TARGA framegrabber digitized pixels into eight bit grey scale creating a quantization error. Its array was only 400 rows X 512 columns (400X512), which represented a region on the display monitor slightly smaller than the entire monitor in both dimensions. Obviously, sampling errors are occuring throughout the detection process. Noise too is being introduced at each step along the way.

Childress and Walrond purchased a charge injection device (CID) camera and framegrabber system that had pixel elements in the framegrabber matched with those in the detector array. This SPIRICON system would clearly eliminate some of the sampling errors, and CID cameras do not suffer from blooming either. The nonuniformity of response would still be a problem; in fact, the provided software includes a bad pixel file for the camera that accounts for some of this. Unfortunately, this expensive system was inoperative at the start of the thesis effort, and was committed to another thesis while repairs were being made. Should it become available for a follow on, it should be noted that software control of the framegrabber is not possible without the purchase of additional custom software from the manufacturer.

Additional information on the CCD camera and the TARGA framegrabber system is available in Appendix B.

3.2.2 MOSLM Limitations. For those uninitiated to MOSLM operation, review of Appendix A is advised before reading this section.

Perhaps the biggest limitation in the operation of the MOSLM is its binary operation. Of course, Javidi does not see this as a limitation, since his work suggests that binary operation in the Fourier plane is superior. However, a grey scale input image device, like a LCTV, might be more versatile. The great advantage claimed by MOSLMs is that their contrast ratio (the difference between an on and an off pixel) is as high as 30 dB. The much cheaper LCTVs (about $100 apiece) are steadily improving, but are not as good. Another student at AFIT, Kenneth Hughes, reported a maximum of 11 dB for the LCTVs used during his thesis [19:41].

The MOSLM used for this thesis had only 128X128 pixels which was also a limitation. Semetex, the manufacturer, reports devices as large as 1024X1024 pixels. The defects in the 128X128 pixel devices available at AFIT (that sell for more than $10,000 apiece), suggest waiting for technological improvements.

The device used in this thesis had two full adjacent rows near the center that were inoperative. It also had a pair of rows that responded to writes to the other. Several individual pixels would be written to when not addressed, while others would not respond to writes. Figure 7 demonstrates the effects of these defects on a test pattern used in the LPCT experiments of chapter IV.

Two 128X128 devices arrived at AFIT in the fall of

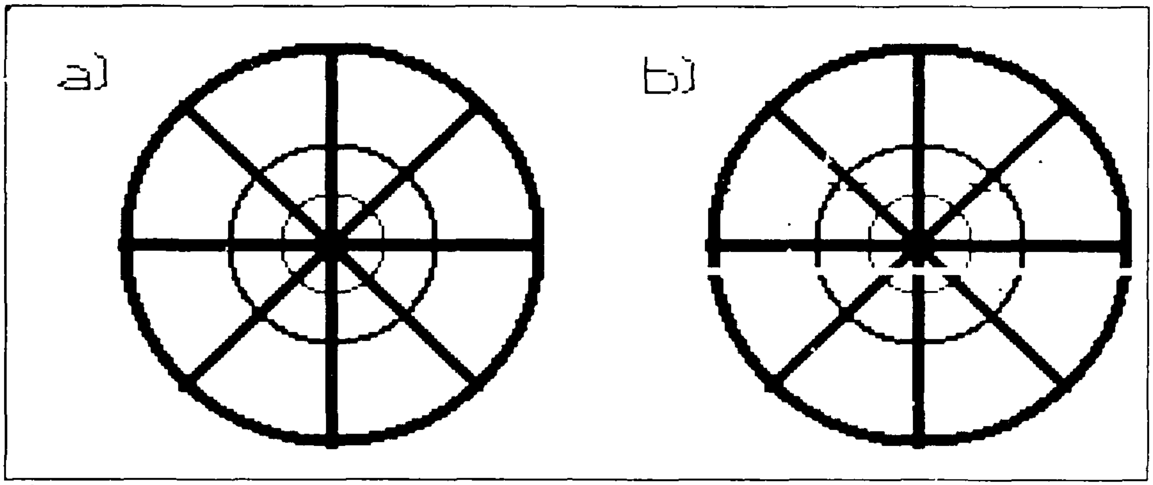

**Figure 7.** Test pattern: a) as intended; b) as displayed on defective MOSLM.

1988. They were factory aligned to write to one pixel at a time instead of the optimal eight pixels at a time. The effect here is that the devices could only operate at less than the maximum 300 frames/second rate. Attempts at alignment of these devices for eight pixel operation failed, so the maximum obtainable, two pixel operation was used.

Shortly after the realignment, one of the devices began to demonstrate eratic behavior and is now unusable. Fortunately upon delivery of the MOSLMs, the manufacturer had acknowledged substandard performance in that array. They delivered a replacement array in the summer of 1989 and work has begun on replacing the defective array. A valuable, but fortunately not costly, lesson has been learned here. To increase the number of pixels that can be written at a time, it is necessary to increase the current to the device. As is the case with most nonlinear optic materials, this risks destruction of the device. Although it is not clear whether or not that occured, one should exercise caution, and wait for the experts at Semetex to improve their technology before being concerned with speed. After all, these are developmental devices. Incidentally, replacement arrays are currently running about $2500.

3.2.3 Input Data. The FLIR data available was provided by the US Army and contained ground vehicles in open fields. This data may be useful for testing scale invariance in the future, but will not provide a realistic rotation testing capability. Future developments might include synthetic aperture radar (SAR) data, or laser range data, amongst others, as candidate applications.

Several frames containing the same targets at the same range were available. These frames were taken at different times of the day, and the targets do not occupy the same pixels in each frame.

The infrared images, in their original form, were 240X640 pixels, each represented using an eight bit grey scale. However, close inspection of the imagery revealed that every other line was lighter than the one adjacent to it. To remedy this, the imagery was reduced by two in both dimensions. This was accomplished by averaging a 2X2 pixel area into 1 pixel; the resultant image would be 120X320 pixels. In fact, the software written to perform this conversion creates a 71X320 pixel file by eliminating some rows above and below the targets. Figure 8 pictures one of these 71X320 grey scale images (8a), as well as various binary images that should clarify the rest of this section.

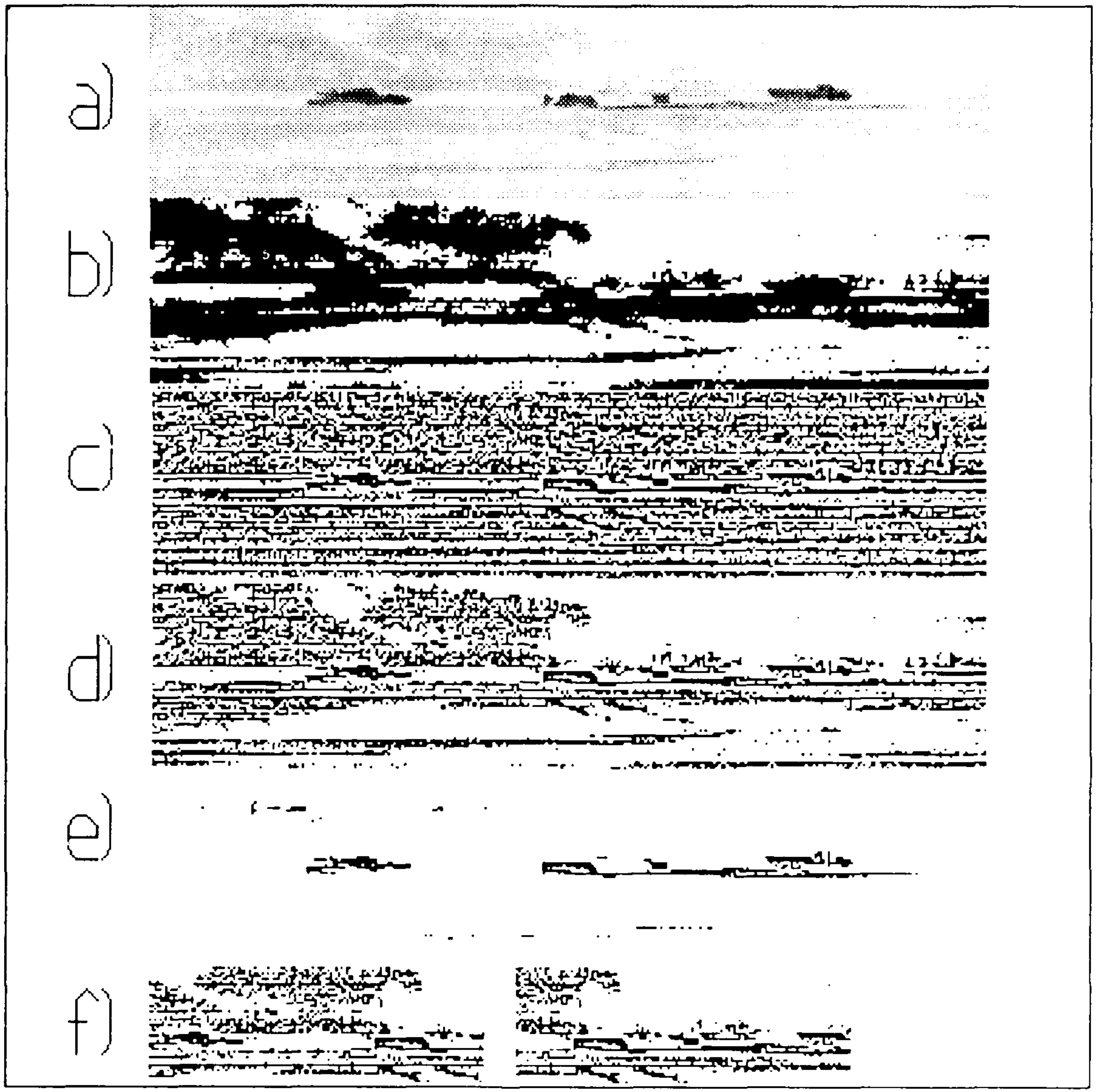

**Figure 8.** FLIR imagery: a) 71x320 image; b) a binarized on scene average threshold; c) a binarized on local 3X3 average threshold; d) c "ANDED" with b; e) a binarized on 1.5 times scene average "ANDED" with c; f) demonstration of subscene overlap.

Next, the images were binarized so that they could be displayed on the MOSLM. The scene average pixel value was used as a threshold for binarization (8b). As a localized adaptive energy normalization technique, a local 3X3 pixel area's average was also used to threshold (8c). Figure 8d was created by performing a pixel by pixel "AND" operation of these two binary images. The resultant scene highlighted pixels that were both brighter than their surrounding area, as well as brighter than the average of the entire scene. Typically, targets are much hotter than their surrounding background, so using a value slightly larger than the scene average in 8b would further reduce scene noise, without deleting many target pixels, improving signal to noise ratio (8d becomes 8e). Although software allowed for variations of this scene threshold (say to 1.5 times the scene average), the experimental results presented later in this chapter will use the noisier, scene average thresholded scenes (8d instead of 8e). This is to demonstrate the correlator's robust performance even under less than ideal conditions. The less noisy scenes give even higher correlation peaks.

The software, located in Appendix C, creates four 45X128 pixel scene files for use in the JTC. This allows for a 64 pixel overlap between files, which ensures that at least one of the four partial scenes will contain the entire target (8f). The reason for the restriction of 45 pixels vertically is explained in section 3.2.4.

Templates were created by hand segmenting binarized scenes with assistance from some software that restricted them to an area of 29X65 pixels. Experimental results include templates from one scene correlated with either the same scene, or a different scene.

3.2.4 Fringe Binarization. Redisplay of the intensity distribution detected in the Fourier plane onto the input device must take into account several issues. The ideal correlation peak will be a delta function (a double delta if both crosscorrelation regions are considered), which means that the redisplayed Fourier plane would ideally be a sinusoid. Unfortunately, real targets in real scenes will not provide perfect sinusoidal interference fringes in the Fourier plane (This isn't Young's double pinhole experiment) and binarization will distort a sinusoid to a square wave at best. Furthermore, the effect of redisplaying the fringes on a pixelized device like the MOSLM will create multiple orders of the correlation plane. These multiple orders may overlap if intelligent display of the input images is not exercised. By restricting the scene to just 45 rows, the maximum separation of scene and template was small enough to avoid this overlap.

Typically, binarization of the detected Fourier plane is accomplished using a simple threshold. Since interference is not generally completely destructive, many relative lows in the fringe pattern can be missed using this technique. This is particularly true of the frequency components of the greatest intensity.

As was mentioned before, the techniques used during this thesis (see Figure 6) can provide improved results. After the scene and template are displayed together and the joint transform is detected in the Fourier plane, the scene alone is displayed and its Fourier transform is detected. The next step is to subtract the two and threshold the difference for binarization. The effect is to highlight pixels with constructive interference and reduce pixels with destructive interference. The simulation presented later in this chapter demonstrates this quite dramatically. Two techniques of thresholding at this point were attempted during experimentation. First, a threshold of zero was chosen, then a threshold of the average difference was chosen. The latter proved to be slightly superior in experimentation.

Two additional benefits were gained from this binarization technique. First, the effect of noise and nonuniformity of response was reduced with the subtraction. Second, the $|S|^2$ term was eliminated which allowed the scene and template to be brought closer together in the original input frame. In fact, it was this latter result that originally motivated trying this technique.

3.2.5 Spatial Frequency Components. Much debate over which spatial frequencies contain the essence or Gestault of an object has occured at AFIT [17]. In general, it is the lower spatial frequencies that usually allow a target to be detected since they are usually of greater amplitude, and the higher ones that allow a target to be classified since they carry the detail of an object.

To ensure good performance from the JTC, the focal length of the Fourier transforming lens (L1 of Figure 2) was chosen to include approximately one full diffraction order on the camera (CCD1). This resulted in the choice of a converging lens with a 500 mm focal length.

Since the MOSLM allowed only 128X128 pixels to be displayed, a 256X256 pixel section was detected and reduced before binarizing. The net effect was that spatial frequencies only as high as about a three MOSLM pixel period ($1^1/_2$ ON, $1^1/_2$ OFF) were detected. The reduction from 256X256 to 128X128 reduced some of the noise and nonuniformity of response effects, but also reduced the clarity of the more closely separated fringes. This reduced clarity had impact on the maximum allowable separation of scene and template in the original input frame. Therefore, this was another driving force that was applied to the ultimate selection of the 45 pixel high scene's location.

One digression is perhaps necessary here. The approach used in this thesis was more akin to a technician making the most of inadequate resources, than that of an engineer pursuing an optimal design. As an example, the choice of the lens focal length was driven more by available lenses on the laboratory shelf than anything else. Many other decisions that seemed appropriate at the time may prove to be erroneous given proper modelling. Likewise, the choice of 45 pixel rows and the scene's location were driven by the bad lines in the MOSLM as well the more technical issues mentioned above. There's nothing magical about 45 rows, it just worked well consistently!

3.2.6 Correlation Plane Interpretation. Obviously the first step in interpreting the correlation plane results is to first detect it. The question at this point is which portion of the plane should be observed. The final decision was to framegrab a 320X256 pixel region and reduce it to 160x128 pixels. This frame reduction was done with noise reduction in mind. As was mentioned before, it may not prove to be the most prudent decision after further analysis is done. This 160X128 pixel region could be divided into three distinct parts, a central autocorrelation and an upper and lower crosscorrelation (see Figure 4 in Chapter II). The peak detection routine searched the upper and lower 50x128 pixel regions for the top ten values in each. These were rank ordered by their grey scale pixel value. (A sneak preview of the figures of section 3.4 may also clarify the ensuing discussion.)

If two peaks on either half were within 5 pixels of each other, they were combined by eliminating the lower valued one from the list, and adding five to the larger one's top ten list value. This eliminated redundant accounting for the same target location while acknowledging the presence of a wide correlation peak due perhaps to imperfect focusing.

Stray reflections, noise, and nonuniformity of response could also cause peaks in one crosscorrelation region that had no corresponding peak in the other. These were accounted for by checking the two top ten lists to ensure that a candidate target location appeared in both lists. With the exception of the number one value, if a peak location in one crosscorrelation was not mirrored by one in the other, it was eliminated from the top ten list.

These two processes did help reduce the number of candidate target locations identified by the correlator.

One issue not discussed yet is that of relating correlation peak location to that of candidate target location. Since alignment might distort the theoretical correlation peak locations, the system was calibrated for peak interpretation. Two small identical annuli, whose locations were known a priori, were used as template and scene objects in the original input frame. The location of their correlation peaks acquired after running them through the JTC were stored and used for calibrating the setup. The scene annulus was successively placed in each of the four corners of the scene space and correlated with the template annulus placed at the customary template center. Subsequent correlations using FLIR images would recall these calibration peak locations and linearly interpolate to relate peak locations to candidate target locations in the original scene.

The final display output of the JTC was the original input frame with an "X" at every candidate target location. The brightness of the "X" on the monitor was determined by the top ten list value for that location. Original binary input frame pixels were displayed as fully bright (Hex FF) or fully dark (Hex 00).

In addition, a scene that was segmented based on the template size and the candidate target locations was also displayed. This segmented scene could be run through the correlator again for better target discrimination, since this segmentation process reduced scene clutter.

### 3.3 Simulation Results

The results of two simulations will be presented in this section. The first is a comparison of the classical JTC with that of a binary JTC using a mean value threshold, and one using the thresholding technique employed in this thesis. All three use binarized FLIR images as inputs. The second is a comparison of classical JTCs employing the four combinations of binary and grey scale scenes and templates. This simulation demonstrates the importance of energy normalization.

The simulations were performed using Imaging Technology image processing equipment made available by the Aeronautical Systems Division (ASD\ENAML). Software to convert TARGA file formats to those of the Imaging Technology equipment and vice versa are in Appendix E.

The Imaging Technology system does not retain DC values in calculating its fast Fourier transform (FFT), and its output is always eight bit grey scale normalized to the maximum value. This may not suffice for some of the more rigorous readers, but does adequately demonstrate the point.

Figure 9 is a comparison of various JTCs that each treat the joint transform plane differently Frame a) is the original input frame with the scene binarized like Figure 8d (the "ANDED" version); b) is its joint transform; and c) is the classical JTC correlation plane (i.e. $|FT|^2$ of b). The correlation peak that locates the target appears as a black dot above and to the right, and below and to the left of center. Frame d) is b binarized on the mean value of the entire frame; and e) is the correlation plane generated by this fringe binarization technique. The peak in e is narrower than the one in c, (usually an indicator of correlator performance, the narrower the better). However, since the Imaging Technology equipment normalizes the output to a maximum value, comparisons of absolute values will not be useful here. Frame f) is the Fourier transform of the scene alone; while g) is b minus f binarized on the sign of the difference. Finally, h) is the correlation plane created by g, which is the technique used for the later experimental correlations using optics. Notice the lack of a large central autocorrelation in h).

Figure 10 contains side views of surface plots for the three correlation planes. Figures 10a and 10b are the views along the vertical and horizontal directions of figure 9c respectively. Similarly, 10c and 10d relate to 9e, and 10e and 10f relate to 9h. The immediately noticeable difference between 9h and 9c or 9e is the lack of a large central autocorrelation in 9h, due to the subtraction of the $|FT|^2$

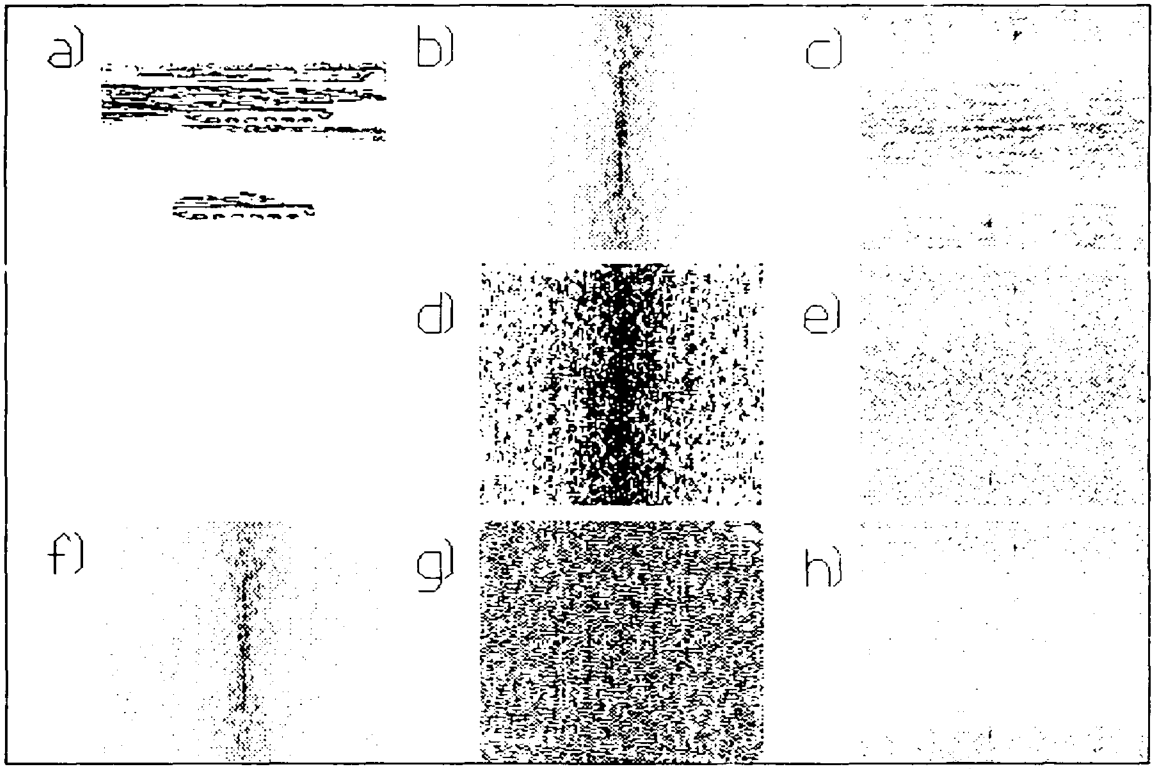

**Figure 9.** Simulation of JTCs: a) input frame; b) joint transform; c) correlation plane from b; d) b binarized on mean; e) correlation plane from d; f) Fourier transform of scene alone; g) Binarized difference between b & f; h) correlation plane from g.

of the scene alone. A second, but perhaps less conclusive (or apparent) fact is that the noise floor about the correlation peak is lowest for 9h.

One explanation for the improvement of 9h over 9e is as follows. Figure 11a shows a cross section of a typical 2D Fourier transform of a scene. Figure 11b shows the same scene joint transformed with a template. Notice that when a threshold line is drawn, some of the fringes both in the high intensity area and the low intersity area can be lost. Figure 11b can be thought of as the interference fringe pattern superimposed upon the scene's bias signal. Figure

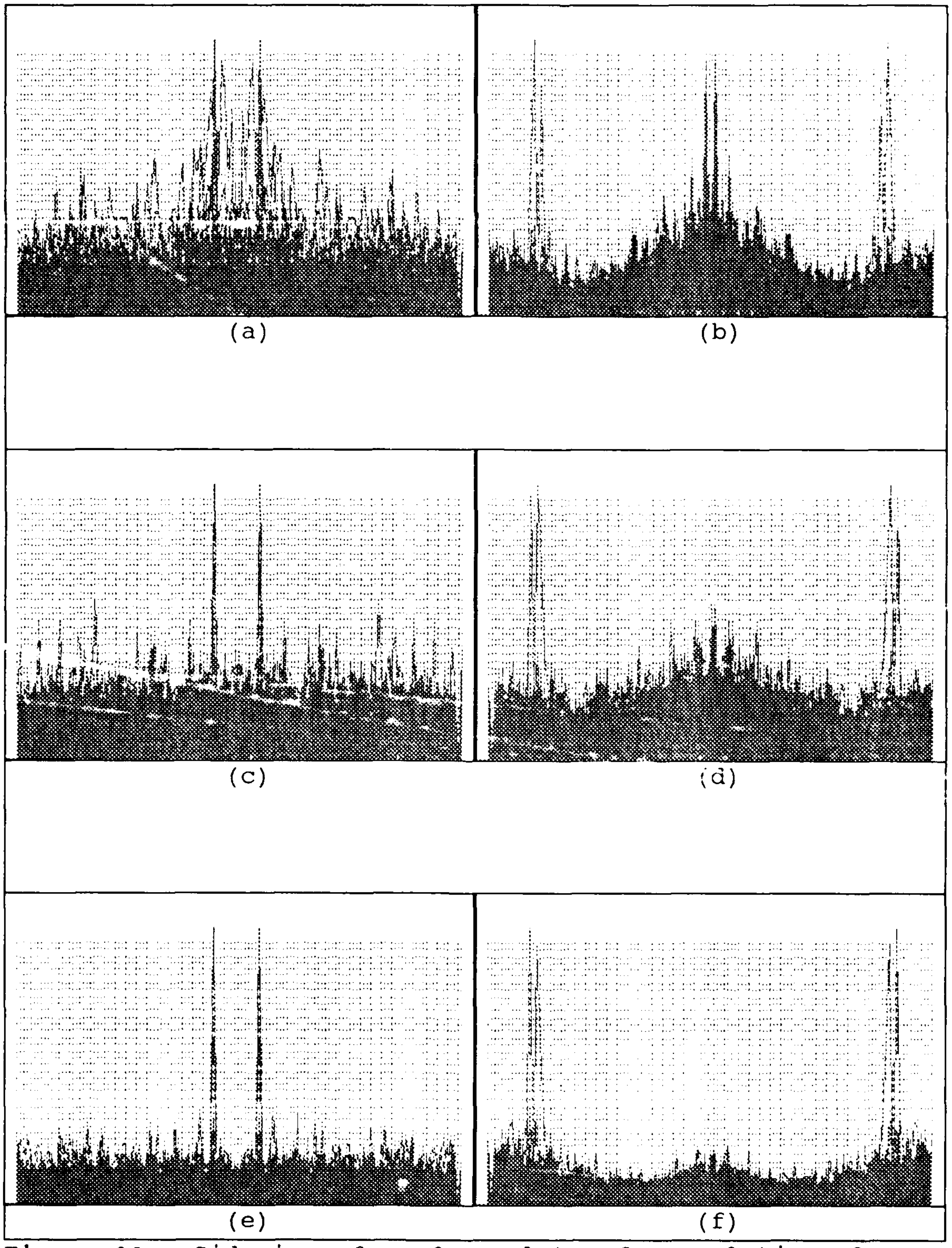

**Figure 10.** Sideview of surface plots of correlation planes in Figure 9: a) along vertical direction of 9c (classical JTC); b) horizontal direction of 9c; c) & d) same as a & b for 9e (simple binary threshold); e) & f) same as a & b for 9h (scene subtracted).

11c shows what happens when the bias is removed by subtracting 11b from 11a. Notice that a threshold will now capture all of the fringes.

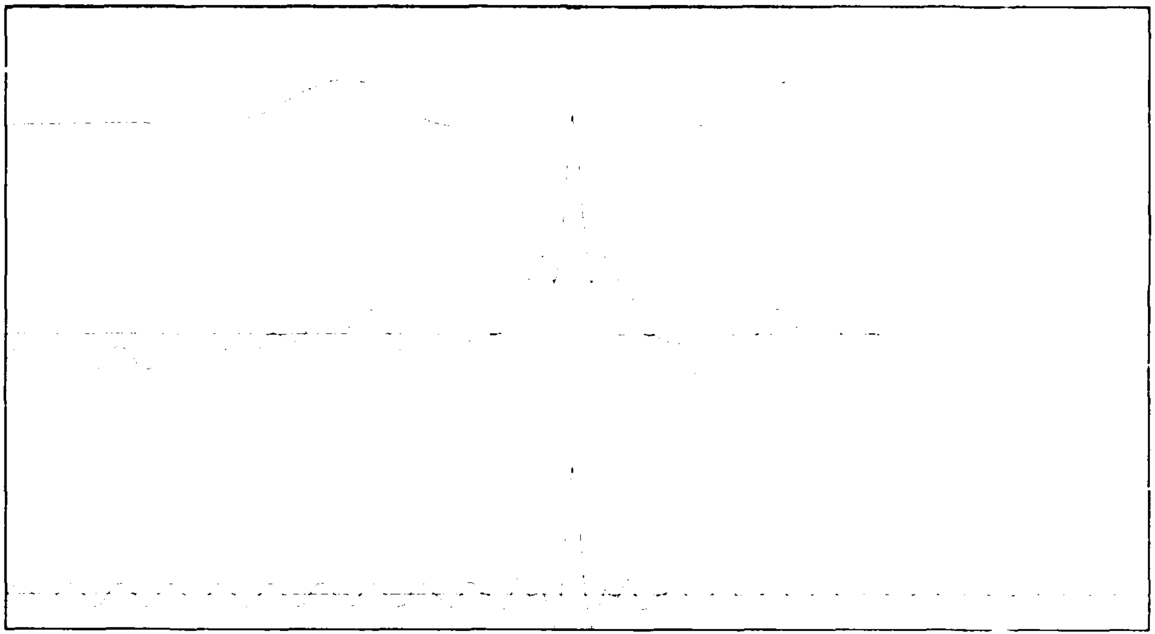

**Figure 11.** Explanation of improved binarization technique. Cross sections of 2D Fourier transforms from top to bottom: a) scene alone; b) scene and template showing one possible very bad simple threshold; c) b minus a showing improved threshold technique.

Figure 12 compares the effects of grey scale versus binary inputs to a classical JTC. It is divided into four columns of three frames. The first row displays the original input frames to a classical JTC. The second row are their respective joint transforms and the third row their correlation planes. The various combinations of greyscale and binary representations are present for both the scene and template. The significant result here is that only in the binary-binary case is their a clear winner for the highest correlation peak which correctly locates the armored personnel carrier. In all other cases, either the tank wins out, or it is a close contender (along with a non target bearing zone roughly in the middle of the scene).

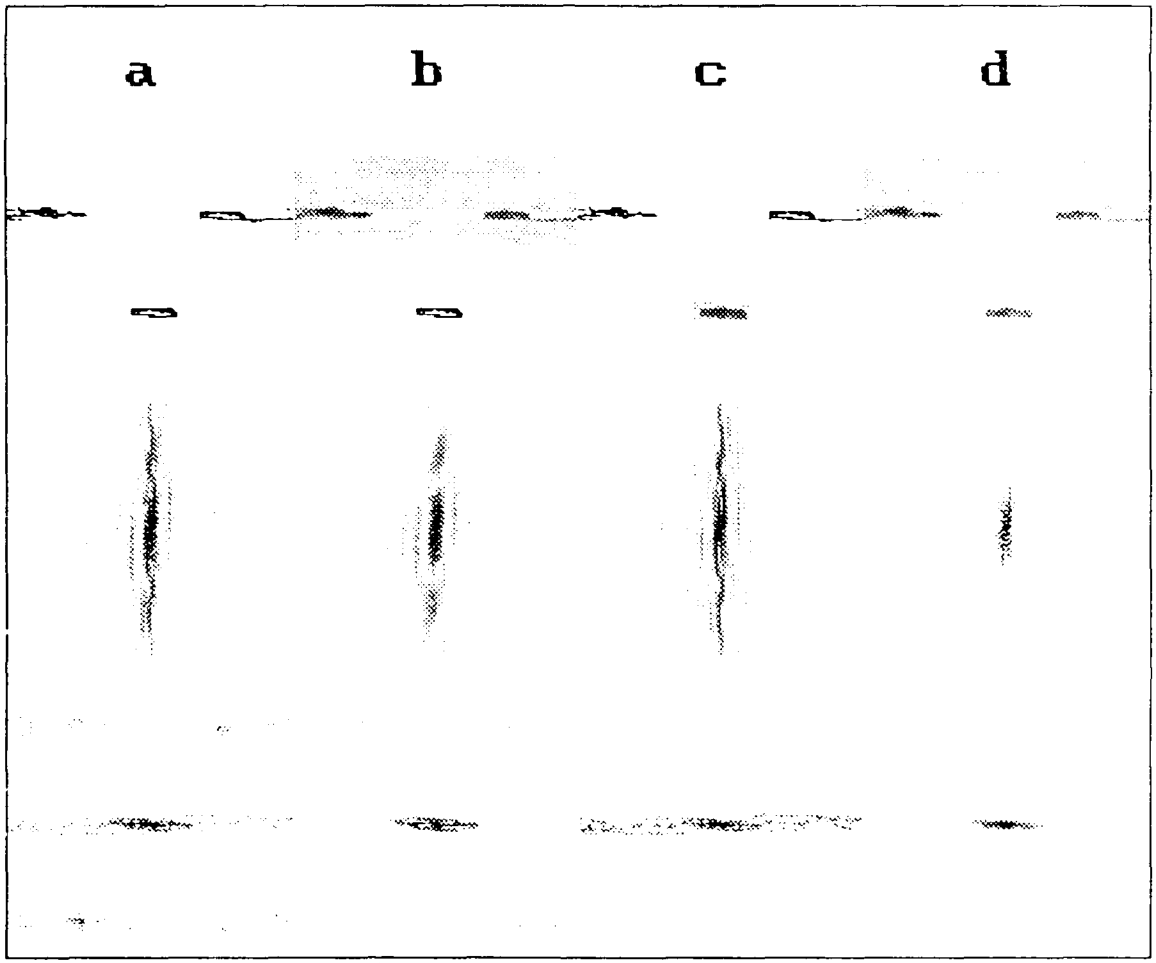

**Figure 12**. Simulation showing input binarization effects on the classical JTC: column a) input, joint transform, and correlation plane for binary scene and template; b) same for grey scale scene and binary template; c) same for binary-grey; d)same for grey-grey.

To make interpretation of Figure 12 slightly easier, Figure 13 is offered. Frames a through d represent side views of surface plots of the four correlation planes. They are along the vertical direction, and include only the top crosscorrelation region of the plane. Note the large peak

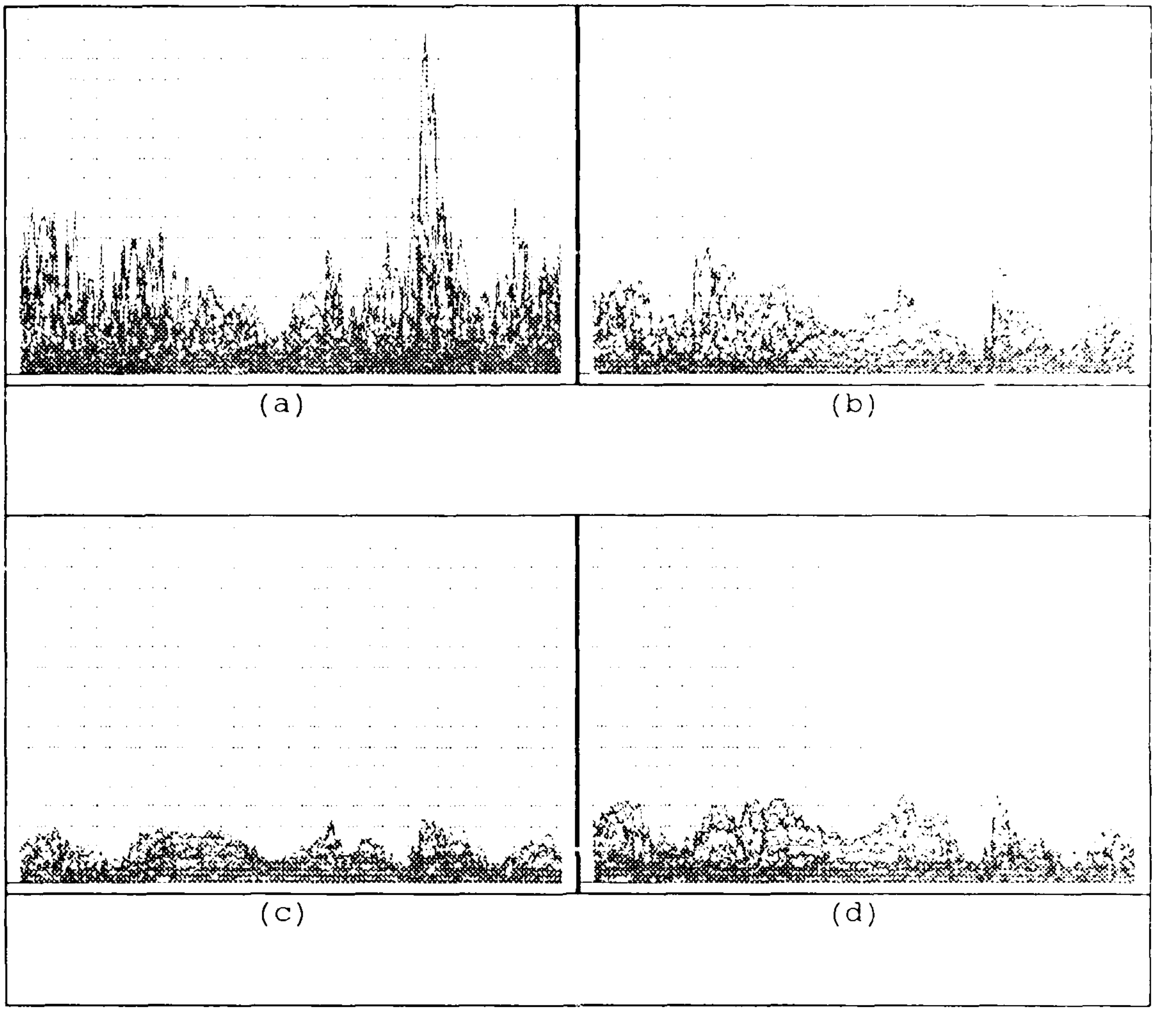

**Figure 13.** Sideview of crosscorrelation region of correlation planes in Figure 12. Notice large peak for binary scene and template inputs in a) only.

on the right hand side for 13a. The peaks in all four frames correspond to the dark spots in the crosscorrelations of Figure 12.

## 3.4 Experimental Results

Only a sample of the many scenes correlated with templates are presented here. In general, they will be used to illustrate various points, not to serve as evidence of the functioning correlator.

The quality of the pictures here is not the best. For that reason, viewing a video tape of these images, that was created and left behind with the thesis advisor Dr. Steven Rogers (Major), may be preferred.

All of the pictures have the same layout, reading left to right then top to bottom: a) is the input frame containing both the scene and image; b) is the optically generated joint transform; c) is the optically generated Fourier transform of the scene alone; d) is the binarized version of the difference between b & c; e) is the optically generated correlation plane; f) is e with the central autocorrelation region removed, peak locations are marked with a "+"; g) is the original input frame with candidate target locations highlighted with "X"s; h) is the input frame after segmentation based on candidate target location and template size; i) is the original FLIR scene before binarization for display on the MOSLM. The first correlation is performed using the technique discovered during this thesis effort. The average difference (as opposed to the sign of the difference like the simulation) between the pixel values of the joint transform and those of the Fourier transform of just the scene was used as a threshold for binarization. The results are pictured in Figure 14.

Figure 15 shows the effects of using just a mean value threshold for binarization on the joint transform plane. Although the Fourier transform of just the scene is

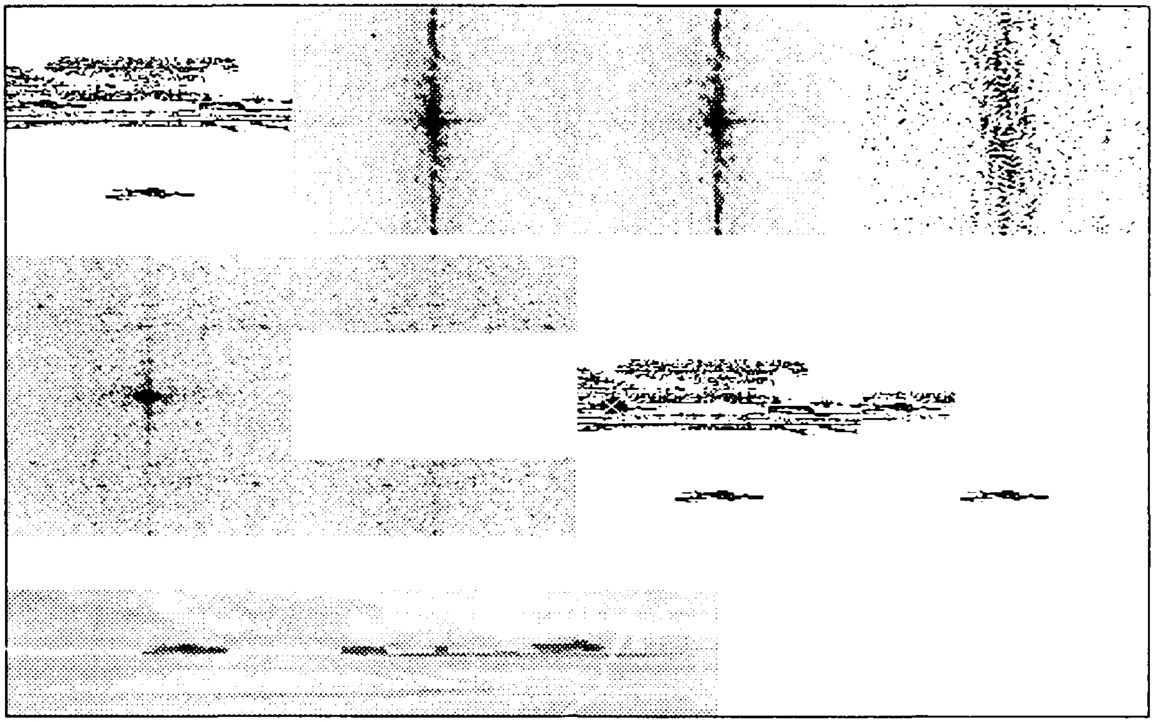

**Figure 14.** Hybrid JTC correlation using average difference for binarization threshold in the joint transform plane. Notice X on target.

pictured, it was not used in the binarization process. Notice that this correlation did not locate the target (no X appears).

Figure 16 is like Figure 15, except that the median value was used for binarization instead of the mean value. This median value was the basis for Javidi's work. Notice again, that the correlator failed to locate the target (the Xs miss the target).

Figure 17 is like Figure 14, except that a zero difference (i.e. the sign of the difference) between the joint transform and the Fourier transform of the scene alone was used, rather than an average difference for binarization. The correlation peak is slightly smaller

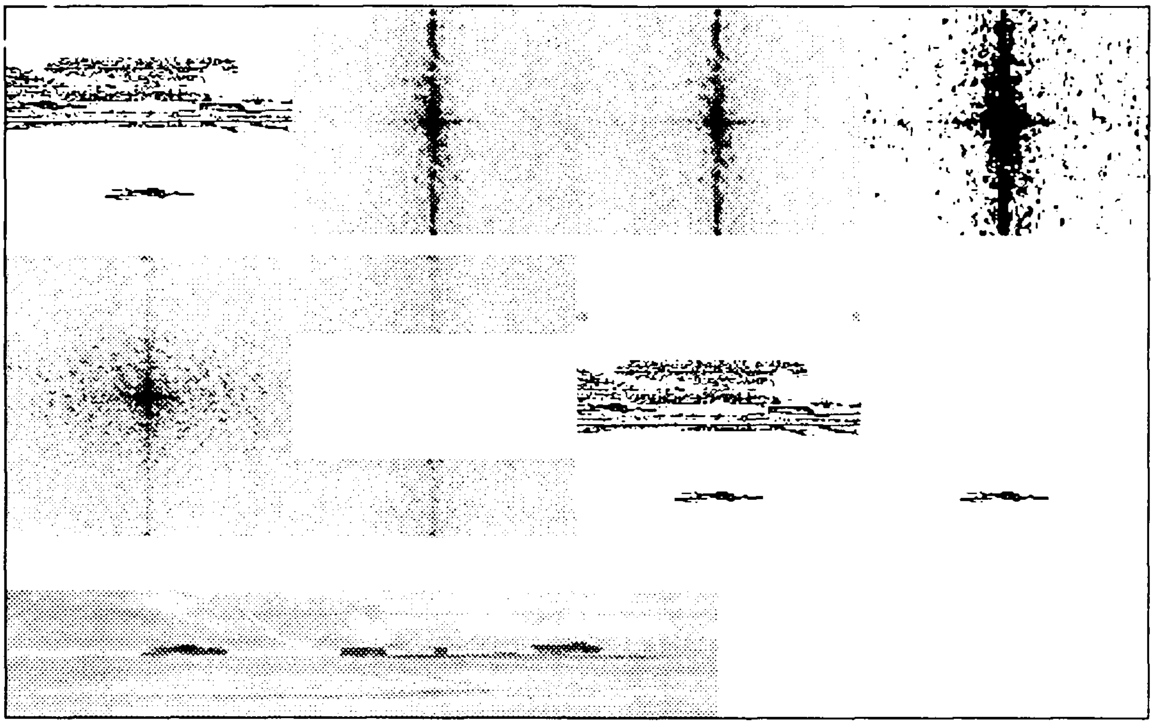

**Figure 15.** Hybrid JTC correlation using the mean value of the joint transform plane for binarization. No candidate targets identifed (no Xs).

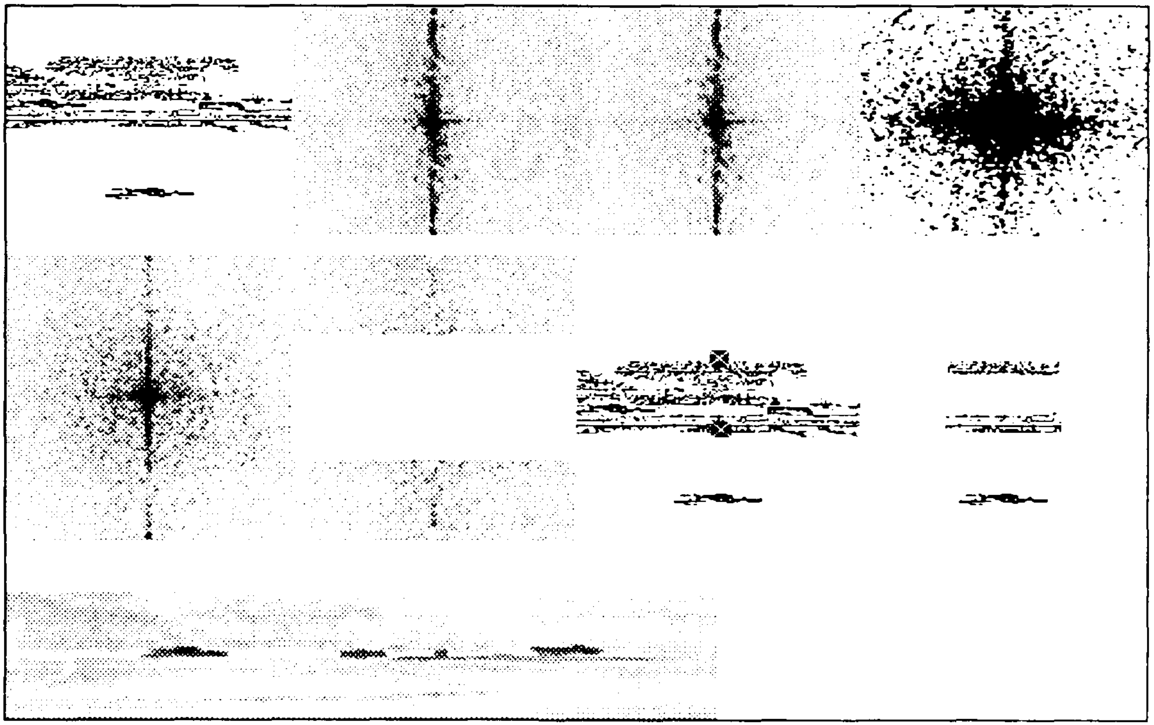

**Figure 16.** Hybrid JTC correlation using the median value of the joint transform plane as a binarization threshold. Notice false alarms and missed detection (Xs appear in wrong spot).

here. The target was still found, but an on-axis bright spot was identified as a correlation peak as well.

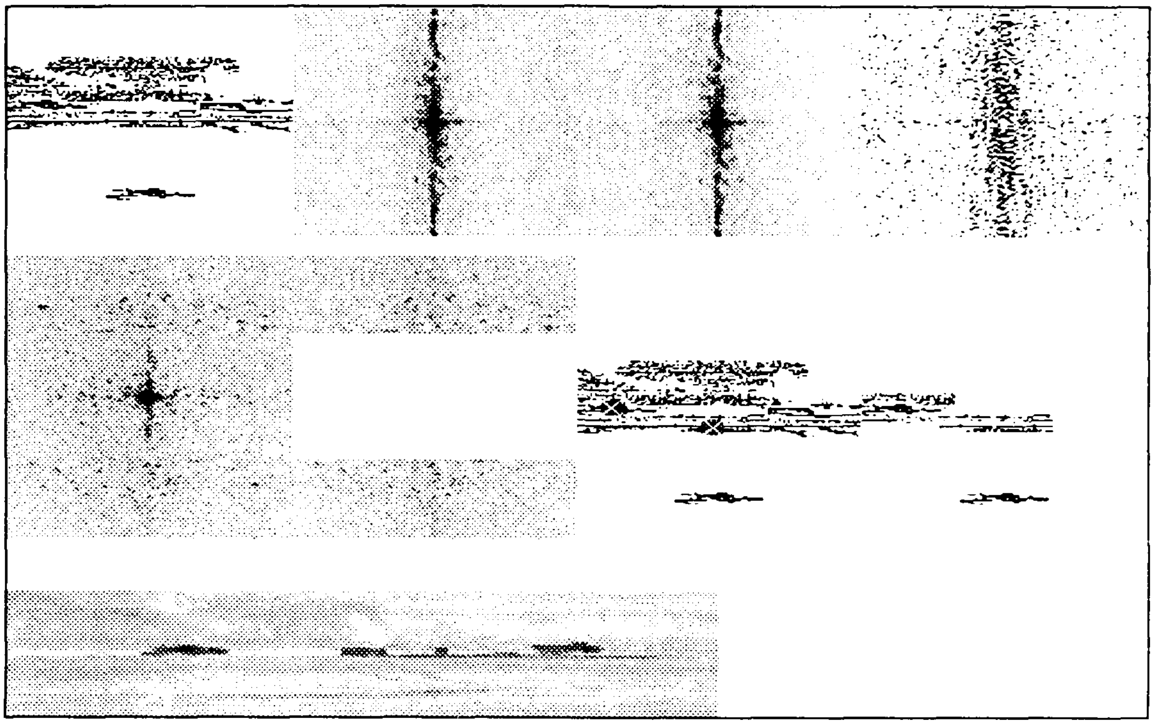

**Figure 17.** Correlation using zero difference as a binarization threshold in the joint transform plane.

Figure 18 demonstrates the correlator's robust performance. Although the template and scene are taken from two different FLIR files, and the target within the scene is highly corrupted relative to the template, the target was still located. The technique of Figure 14 is used.

Figures 19 and 20 demonstrate the usefulness of the segmentation capability. With less than optimum alignment and using only a zero difference threshold, the correlation of Figure 18 produced more than one target location. Using the segmented scene created by this pass through the JTC as input to a second pass, the correlation of Figure 19 results. Notice that further segmentation of the scene is

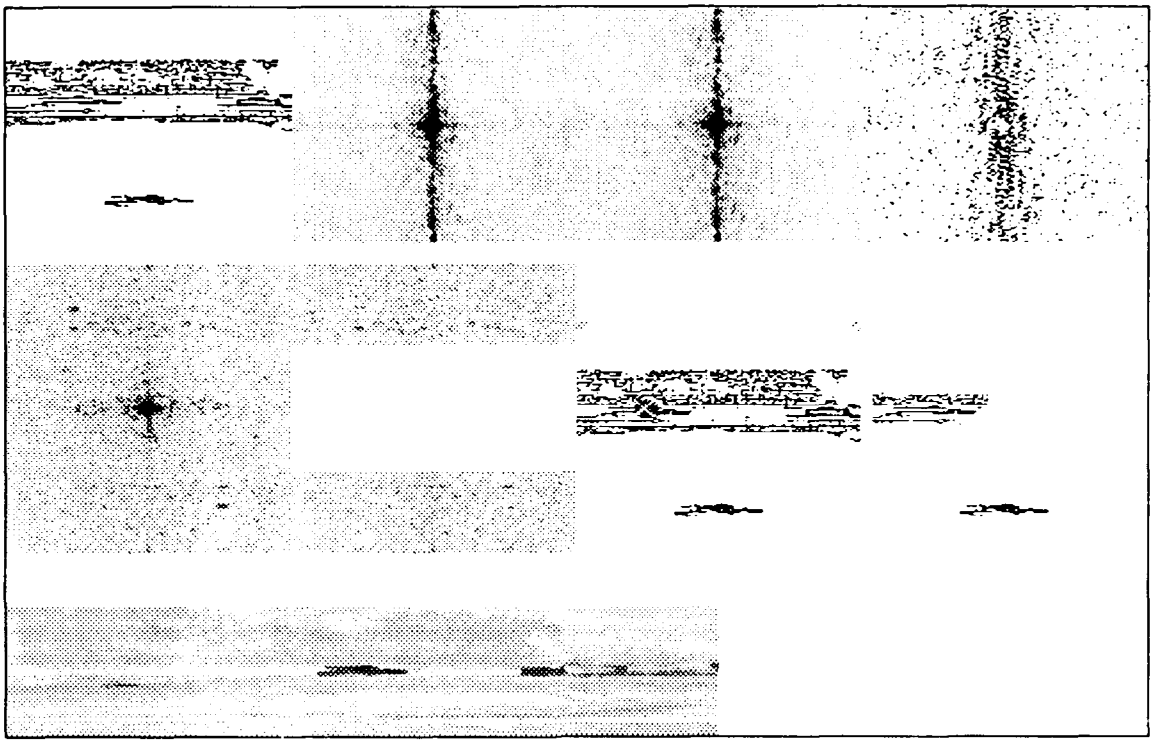

**Figure 18.** Correlation of scene and template, not of same FLIR file origin.

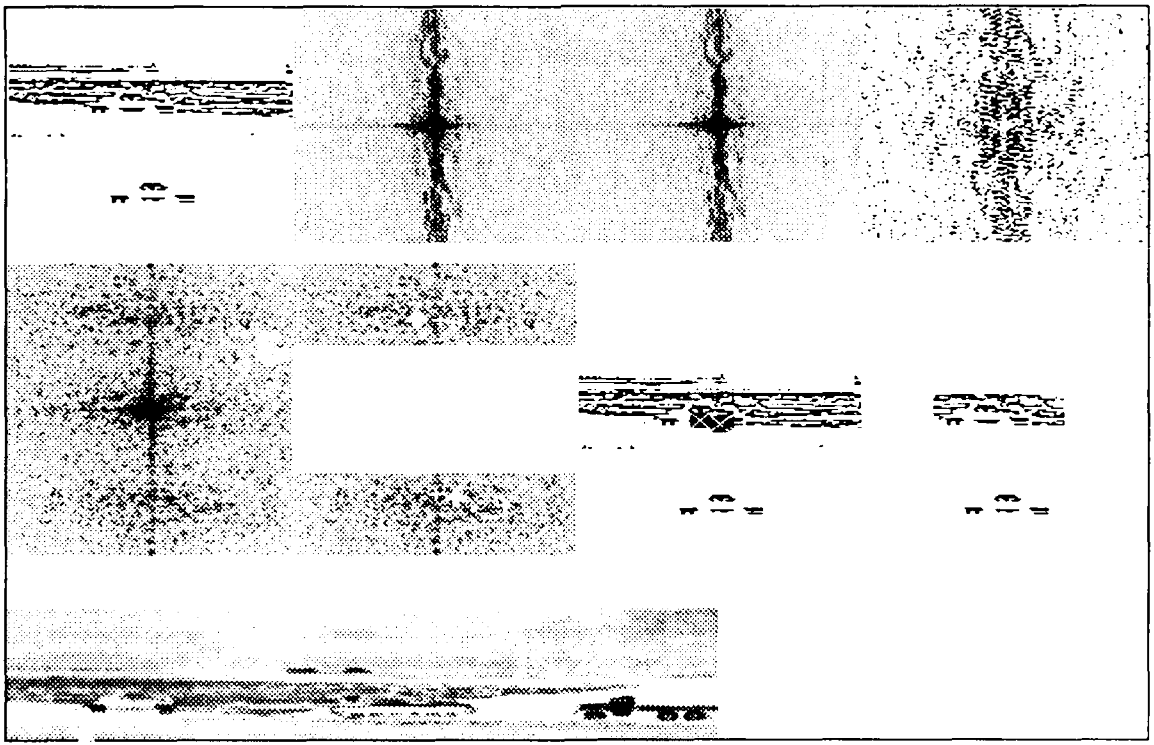

**Figure 19.** Correlation using zero difference threshold for binarization in joint transform plane. Less than optimum alignment.

provided. Classification may now be more accurate due to this segmentation. A third iteration may also be possible.

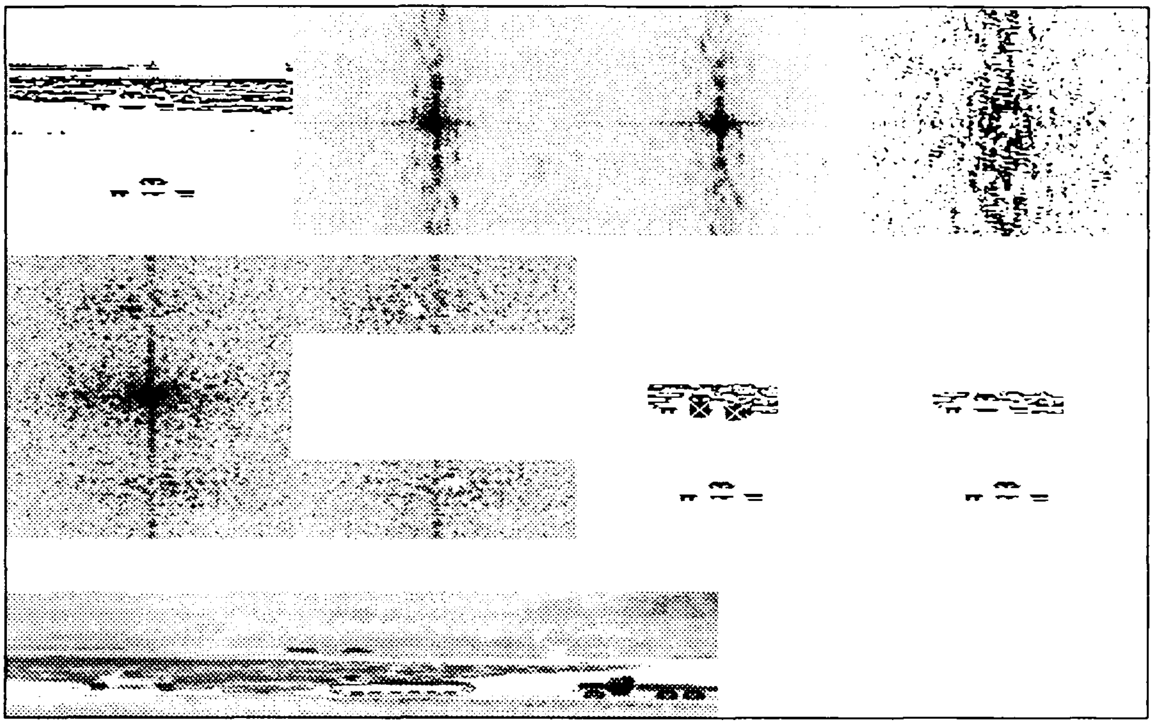

**Figure 20.** Correlation using segmented scene as input.

Figure 21 demonstrates the possibility of locating partially occluded targets. Notice that even though the entire truck is not present, a strong correlation peak is still returned. The effects of blooming (the horizontal line), and sampling error (the vertical fringes) can also be seen in the joint transform.

Finally, Figure 22 shows a scene with a target with a slight out-of-plane rotation relative to the template. Note that the wheels of the tank are not as pronounced either. Despite both of these, the target was still recognized by the correlator as a potential target. This is an important result if a correlator is to be used as a tracker, since

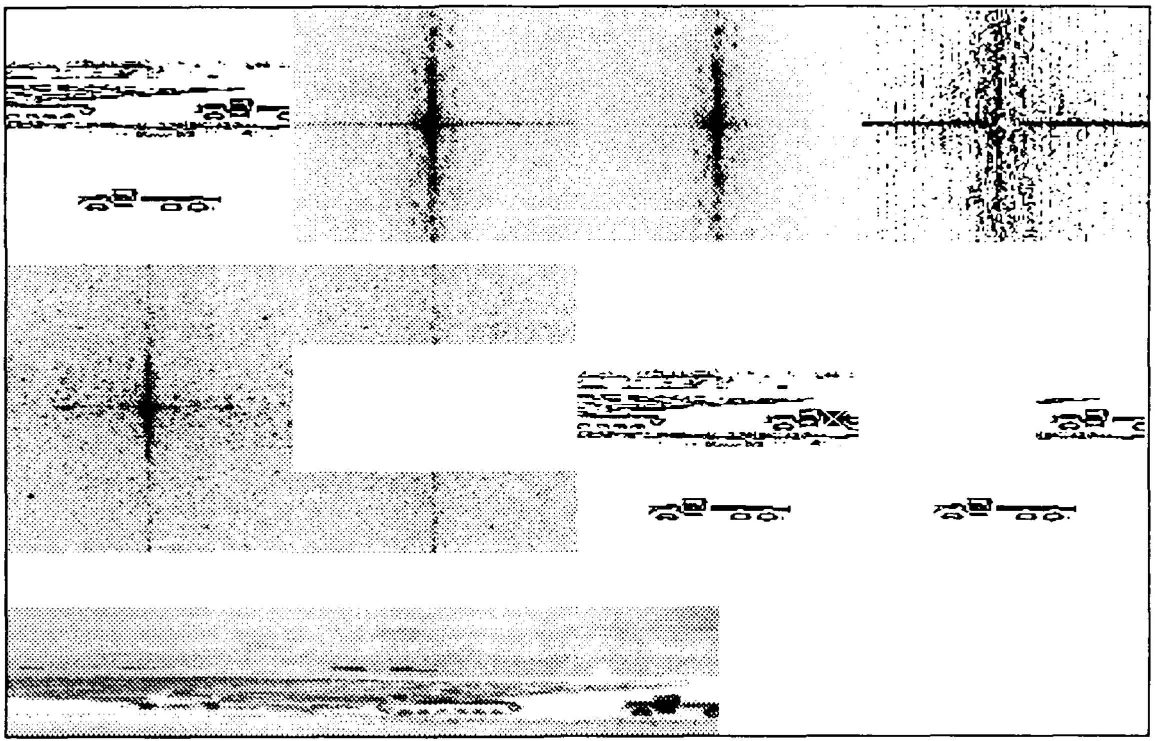

**Figure 21.** Correlation demonstrating partially occluded target.

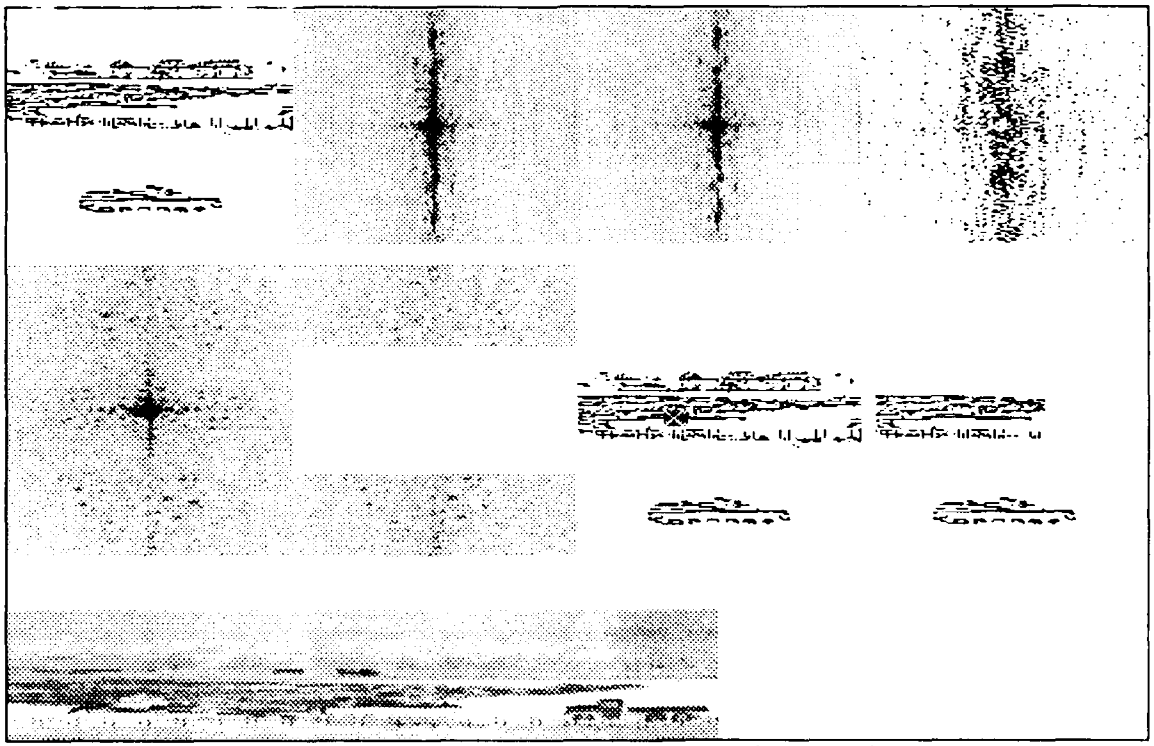

**Figure 22.** Correlation with target having slight out-of-plane rotation relative to template.

slight out of plane rotations are highly likely with moving targets.

## 3.5 Summary

This chapter has presented information concerning the JTC used for this thesis. History, theory, hardware software, simulations, and experiments all have been discussed. A new technique for fringe binarization, discovered during this thesis effort, was offered as an improvement on published JTC designs. A binary mask for FLIR imagery was also offered that provides local energy normalization, thereby improving correlator target detection performance.

Chapter IV will provide theory and experimental results concerning the optical LPCT using CGHs. Chapter V will then cover the use of both the JTC and the CGH in the hybrid architecture.

# IV. LOG POLAR COORDINATE TRANSFORMATION (LPCT)

This chapter will present information about the optical implementation of the LPCT. This thesis was directed only toward incorporating the already existing CGHs, made available from Perkin-Elmer and Mayo's earlier thesis effort [8]. For that reason, only a brief discussion of theory and manufacturing processes are offered here. Further details may be found in other sighted sources.

Following the theory and manufacturing processes sections, some hints on optical techniques will be given. Finally, experimental results for the two CGHs will be compared.

## 4.1 Theory

Image processing through spatial filtering is a common practice in the world of optics. By proper placement of transmissive or opaque objects in the focal (or Fourier) plane, one can modify the modulation transfer function of an imaging system and thereby alter the resultant image.

The field of holography has allowed optical engineers to extend this image processing technique to alter the phase of an object's spatial frequencies as well as the amplitude of these Fourier components. Used as a spatial filter, a hologram would increase or decrease the path length of light passing through certain locations in the Fourier plane with respect to other locations. This alteration of path length is tantamount to a phase modulation of the wave front passing through the plane of the hologram.

As it turns out, this sort of phase modulation through holography need not be performed in the Fourier plane alone. In fact, many have shown that with the help of a converging lens, and by placing the proper phase function in the object plane, one can remap the light amplitude in the object plane from x-y coordinates to log-polar coordinates (ln(r)-$\theta$).

Figure 23 shows the optical arrangement employed by Mayo to perform this LPCT [8:19]. The CGH is placed immediately behind an input transparency that is illuminated by a collimated beam. A converging lens is placed exactly one focal length beyond the input plane, and in front of the output plane where a CCD camera is located.

If the input object is a transparency illuminated by a uniform plane wave, then light leaving the object can be expressed by the real function:

$$g(x_i, y_i) \tag{9}$$

If the CGH modulates only the phase of the input, it can be expressed as:

$$e^{j\Phi(x_i, y_i)} \tag{10}$$

The converging lens will create the Fourier transform of the input in the output plane. So the light at the CCD can be expressed as:

$$G(x_o, y_o) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x_i, y_i)\, e^{j\Phi(x_i, y_i)}\, e^{-j(2\pi/\lambda f)(x_i x_o + y_i y_o)}\, dx_i dy_i \tag{11}$$

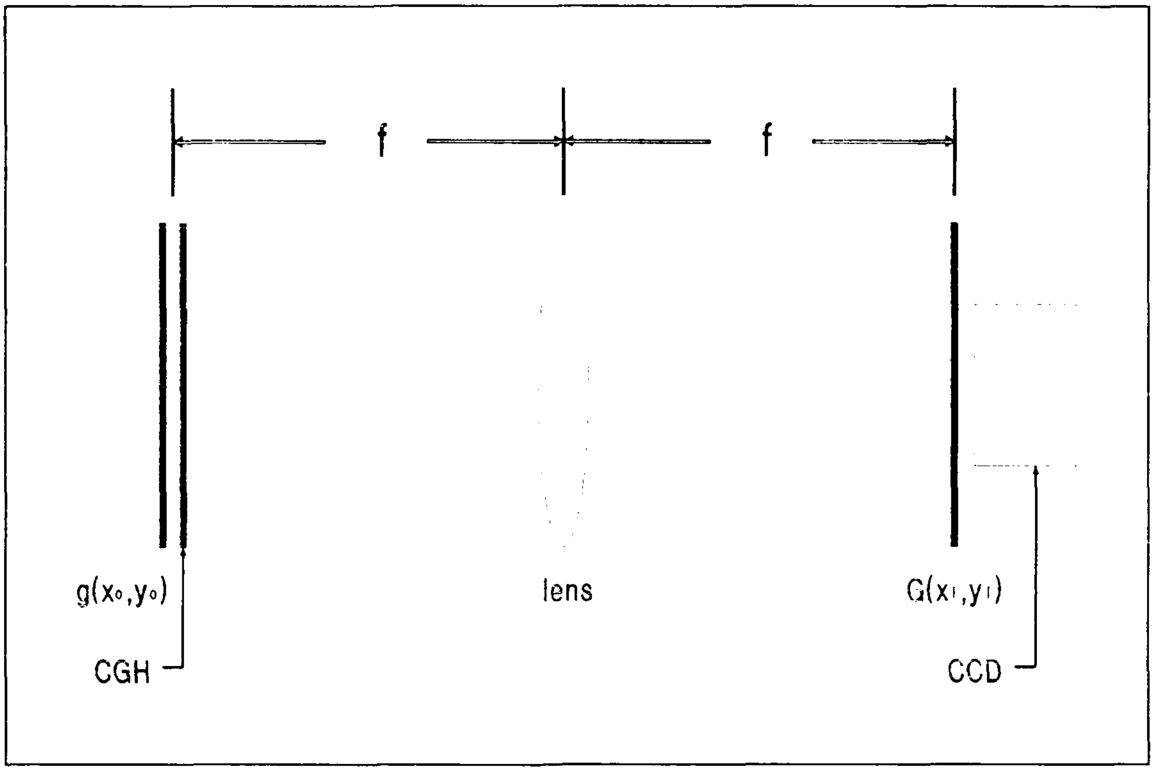

**Figure 23.** Mayo's LPCT optical arrangement.

where:

$$x_o = \ln(x_i^2+y_i^2)^{1/2} = \ln(r_i) \tag{12}$$

$$y_o = -\tan^{-1}(y_i/x_i) = \theta_i \tag{13}$$

It can be shown [20:3099-3104] that the equation for $\Phi(x_1,y_i)$ that will perform the log-polar mapping is:

$$\Phi(x_i,y_i) = (2\pi/\lambda f)[\ln(x_i^2+y_i^2)^{1/2} - y_i\tan^{-1}(y_i/x_i) - x_i] \tag{14}$$

An excellent treatment of this theory, as well as a summary of various holographic techniques is presented in Appendix A of the dissertation of Andrew J. Lee from Carnegie Mellon University [21:153-192]. His appendix would be helpful to those interested in learning more about this subject.

## 4.2 Manufacturing Processes

A number of techniques can be employed to create the $\Phi$

function described above. The Perkin-Elmer CGH was created using a technique known as electron beam (E-beam) lithography. The Mayo-Hill CGH was manufactured using a technique known as transmittance coding using a carrier which can create an "interferogram." Other possible techniques include detour phase holograms.

E-beam lithography is a commonly used technique in the world of very large scale integration (VLSI) circuits. It can also be used to etch a glass sample to produce thickness variations across its surface. These thickness variations, which relate to variations in optical path length, can be made to correspond to the desired $\Phi$ function. The Perkin-Elmer sample, created by this technique, was etched to provide four different thicknesses, as opposed to the more common two thicknesses [22].

An "interferogram," or synthetic binary hologram, consists of a transparent material that has opaque lines drawn upon it. The lines are drawn to model the interference fringes that would be created if an object wave carrying the $\Phi$ function were interfered with an off axis reference wave. Illumination of this interferogram with an on axis plane wave would produce an off axis object wave carrying the $\Phi$ function. The Mayo-Hill CGH was produced by drawing the fringes using a laser printer, then photoreducing this output onto a glass plate [8:80-84].

## 4.3 Optical Techniques

The first optical technique employed was the use of two lens imaging to display input objects onto the CGH [9:27-28]. Since both the MOSLM and the CGH had finite thicknesses, having them share the same plane would be impossible.

A particularly helpful tool created during this thesis effort was a MOSLM test pattern used to qualitatively evaluate the performance of the LPCT process. This pattern, pictured in Figure 7a of chapter III, consists of three circles whose inner and outer radii are logarithmically related, and 8 radial lines (every 45°). The resultant LPCT should resemble a grid pattern like the one pictured in Figure 24. In this, and all other pictures, increasing ln(r) is downward on the page. A crown shape pattern on the bottom is due to the fact that tne MOSLM has a square display area. On top is a lower intensity mirror image.

On first attempts to use the CGH, some problems arose. The first was that since the MOSLM was a pixelized device, several diffraction orders appeared (see Figure 25). By placing an iris at the focal plane of the two lens imaging system, the higher diffracted orders of this sampling were eliminated (see Figure 2 of Chapter I). Keep in mind, however, that some image sharpness is lost from this.

A second problem was that of camera saturation. Figure 26 shows the results of not placing neutral density in front of the camera.

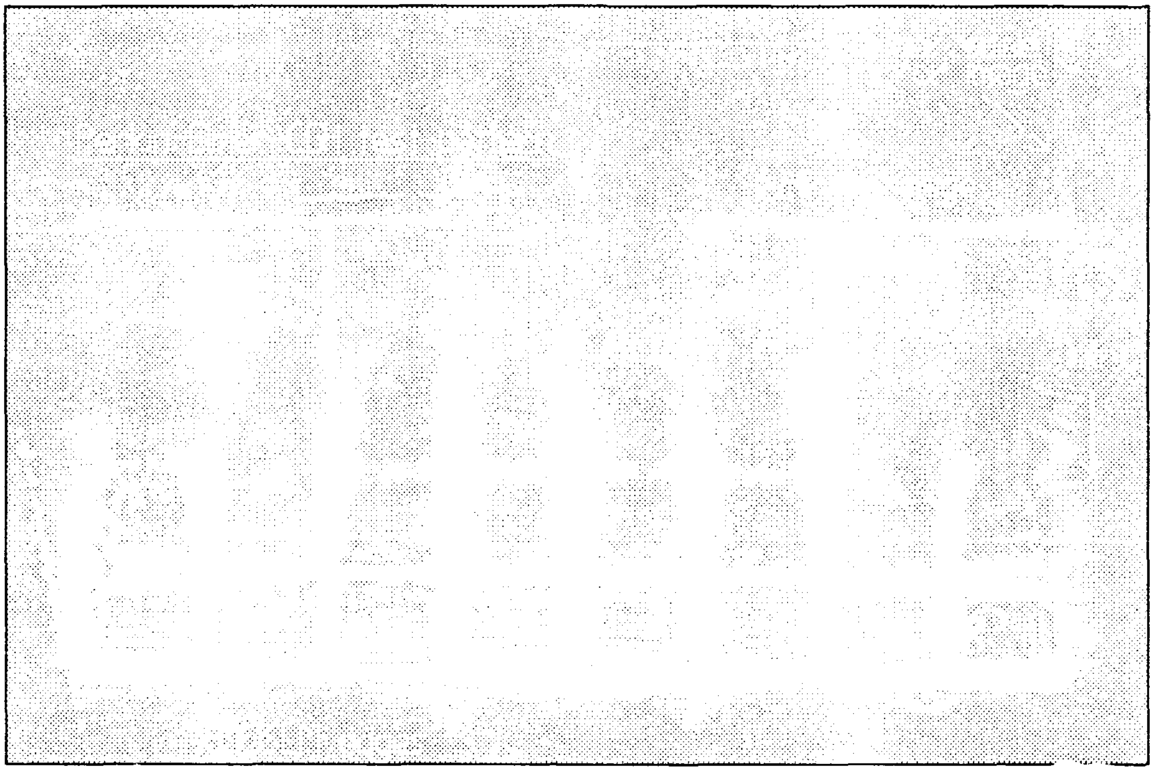

**Figure 24.** Log-polar coordinate transform of test pattern using Perkin-Elmer CGH; Appropriately filtered.

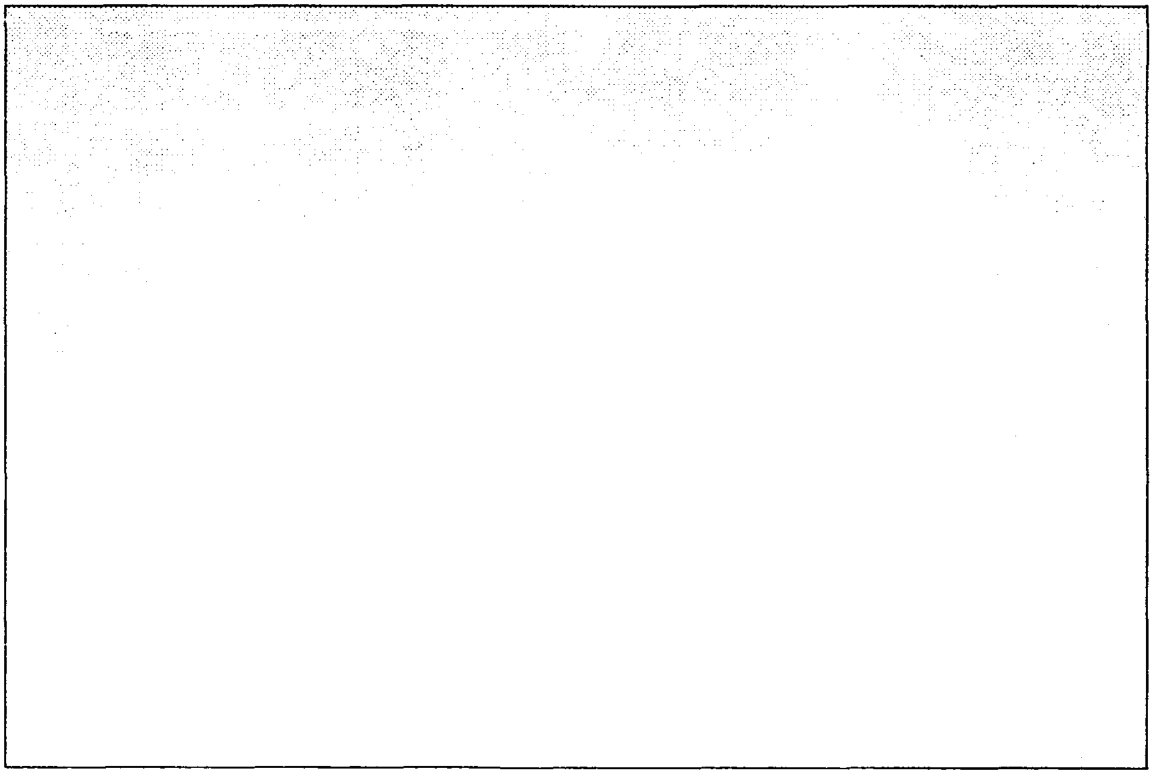

**Figure 25.** Log-polar coordinate transform of test pattern using Perkin-Elmer CGH; No spatial filtering is employed.

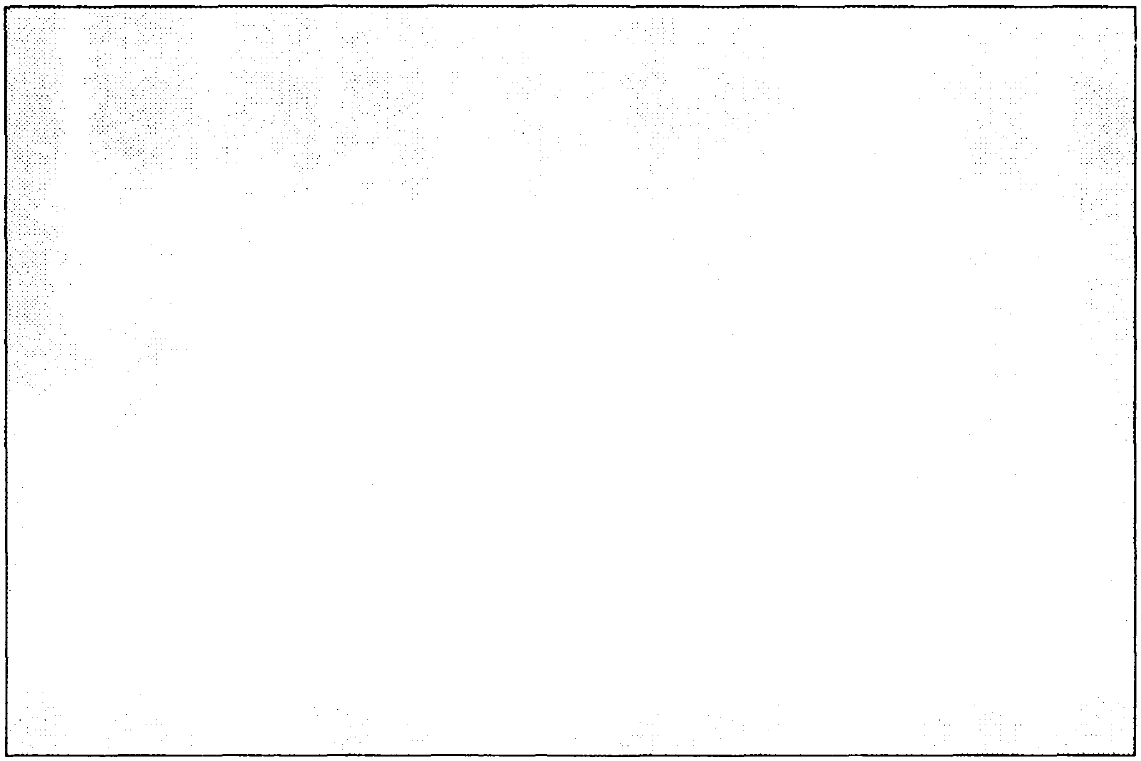

**Figure 26.** Log-polar coordinate transform of test pattern using Perkin-Elmer CGH; no neutral density employed.

A third problem was one of scale. As it turns out, for both of these CGHs, to change the scale of the LPCT, one need only vary the focal length of the Fourier transforming lens used (L4 of Figure 2 in Chapter I) [23]. For the Perkin-Elmer CGH L4 was 86 mm, and for the Mayo-Hill CGH, it was 250 mm that allowed for maximum use of the CCD2 array.

A fourth problem was that of extraneous light distributions. Both CGHs produced on axis Fourier transform "artifacts," as well multiple reproductions of the LPCT. The Fourier transform artifacts appear as bright spots on the LPCT image. Both these and the multiple reproductions could simply be ignored by avoiding these areas during the framegrabbing process.

## 4.4 Mayo-Hill vs Perkin-Elmer

So far only the results using the Perkin-Elmer CGH have been shown. Figure 25 displays the LPCT of the test pattern for the Mayo-Hill CGH. Clearly, the results of the Perkin-Elmer E-beam CGH sample are far better than those of the Mayo-Hill interferogram sample. However, caution should be exercised before judgement is made. Only through a thorough understanding of all of the issues surrounding these two techniques can proper conclusions be drawn. Some observations are offered, however.

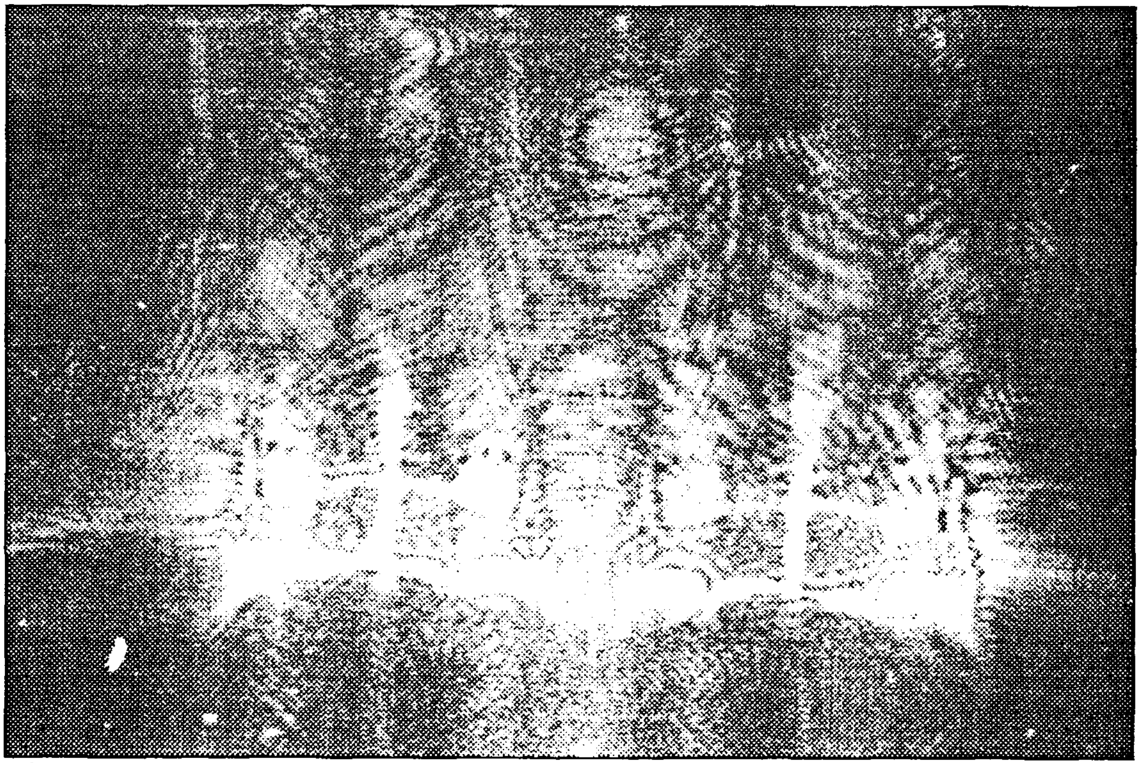

**Figure 27.** Log-polar coordinate transform of test pattern using Mayo-Hill CGH; appropriate neutral density and spatial filtering employed.

James Logue at Perkin-Elmer has suggested that their method will place about 90% of the total incident energy into the primary LPCT. He claimed that the interferogram technique typically places 10% at best into two LPCTs above and below the axis [22].

Both CGHs had lower intensity "mirror image" LPCTs that overlapped the LPCT of interest. In fact, the Perkin-Elmer sample created a low intensity larger scale one in mirror that, unfortunately cannot be seen in Figure 22 (see video tape with Dr. Rogers).

Both CGHs had some trouble in maintaining sharp lines for both the circles and the radials. However, the Mayo-Hill CGH was not nearly as good in this area. The sharpness of the lines appeared to be a stronger function of radial direction than of radius.

Not a defect, but certainly worth mentioning, was the fact that the circle of smallest radius was lowest in intensity, even though it covered the same amount of area as the others in the log-polar coordinate space. This is due to the fact that the larger the circle, the more pixels it contains, therefore, the more total energy it will have.

This will lead to a tradeoff in the selection of neutral density. If a displayed object has a large amount of its energy at large radii, then the risk of not enough neutral density is possible. On the other hand if most of the energy is at small radii, the risk of too much neutral density is possible. Without a priori knowledge of the displayed object's energy distribution, optimum choice of neutral density to ensure square law operation of the detector array may be impossible.

## 4.5 Summary

This chapter briefly discussed the theory of optically implementing the LPCT. Also briefly discussed were the manufacturing processes for creating the two CGHs used for this thesis. Some laboratory techniques were suggested and the experimental results for the two were qualitatively compared.

The Perkin-Elmer E-beam CGH sample clearly outperformed the Mayo-Hill interferogram, both for total energy delivered, and sharpness of the image.

The next chapter will present results of the hybrid optical/digital architecture employing the Perkin-Elmer CGH.

# V. HYBRID OPTICAL/DIGITAL ARCHITECTURE

This chapter discusses the architecture designed and implemented for this thesis. This architecture may provide position, scale and rotation invariant pattern recognition. First, the architecture will be described. Then, engineering considerations will be discussed. Finally, experimental results using geometrical test objects will be presented.

## 5.1 Hybrid Architecture

Figure 28 shows the hardware configuration, and Figure 29 is a flow diagram for the technique used. Refer to these for clarification of the following discussion.

Using the MOSLM as the only input device to the optics, the hardware configuration of Figure 28 can perform two separate functions. Using the beam splitter (BS), light distributions displayed on the MOSLM travel the optical paths that end at CCD1 and CCD2 simultaneously. At CCD1, the magnitude squared of the Fourier transform of these distributions can be detected. While at CCD2, the magnitude squared of the coordinate transformation can be detected as discussed in Chapter IV. A third process, joint transform correlation, is performed as described in Chapter III using the path of CCD1.

The PC performs all equipment control, data storage, and intermediate calculations. Steps 3 and 6 of Figure 29

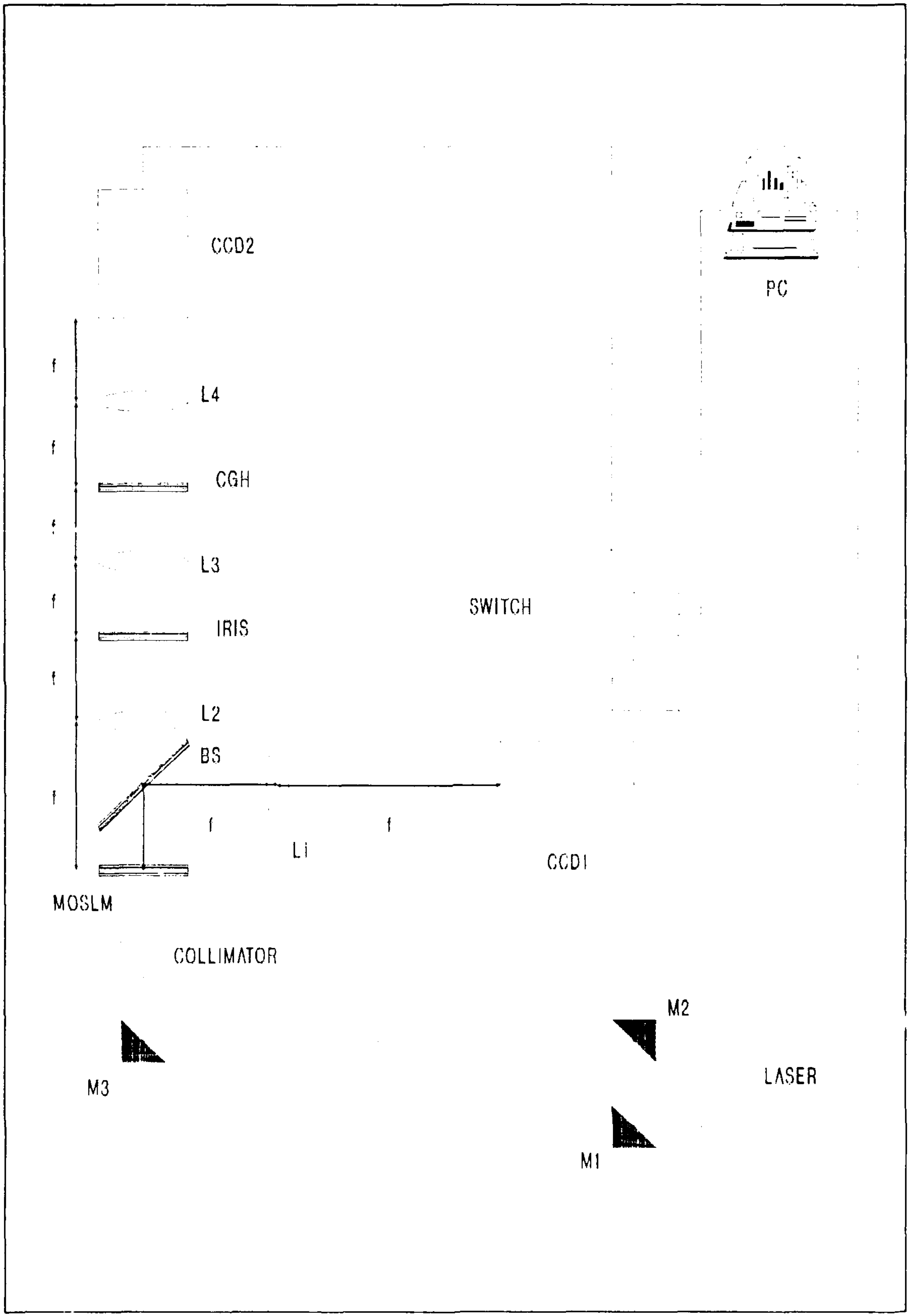

**Figure 28.** Hybrid Optical/Digital Architecture

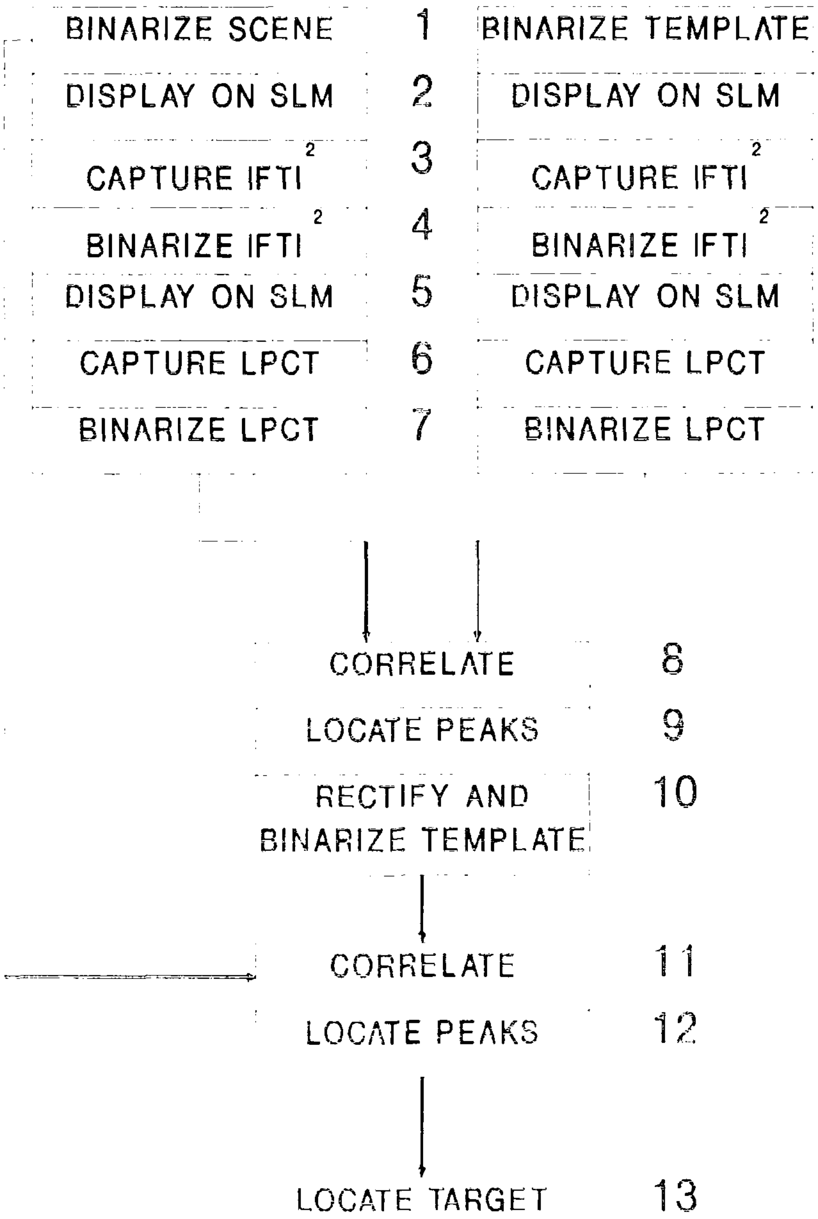

**Figure 29.** Hybrid optical/digital architecture flow diagram.

are accomplished using the framegrabber with CCD1 and CCD2 respectively. In order for step 6 to work correctly, the operator must throw the manual switch, so that CCD2 is connected to the framegrabber. After step 6, the switch should be returned. Future implementations may include automation of this step.

Step 8, correlation of the LPCT-$|FT|^2$ features of the scene and template, is performed using the JTC described in Chapter III, which requires the optics of the CCD1 path. Step 9 was performed by the PC, however software for relating these peak locations to scale and rotation still needs to be written. Due to time and equipment constraints this thesis effort ended with this step. Observations through this point will be explained in sections 5.2 and 5.3.

The locations of these peaks are to give the proper scale and rotation of the target object within the scene as described by the Horev algorithm [4]. Using this information, the template can be rectified to reflect the proper scale and rotation in step 10.

Steps 11 through 13, relating to the correlation of the properly scaled and rotated (rectified) template with the original scene, can already be performed by the JTC discussed in detail in Chapter III.

## 5.2 Engineering Considerations

A number of engineering decisions had to be made to implement the first seven steps of Figure 29. Discussion of these decisions will generally follow the order of the steps, however, some decisions encompassed several steps.

5.2.1 Binarize Inputs. Since only geometrical test objects were used, no decision at step 1 was necessary. However, this may be important when this architecture is tested with real data.

5.2.2 Display Inputs on SLM. To minimize the effects of the optical system's varied distortion from different parts of the input plane, both template and scene were displayed using the same area of the MOSLM in step 2.

5.2.3 Capture $|FT|^2$. Capture of the $|FT|^2$ in step 3 required some thought. The important question here was: What spatial frequency range is important for discrimination? This question was raised before by Kobel and Martin [5:31-33].

After capture, the frame would have to be displayed on the MOSLM. Capturing a 256X256 pixel region and reducing it to 128X128 would allow for higher spatial frequencies to be displayed (up to about a 3 MOSLM pixel period or $1^1/_2$ pixels ON $1^1/_2$ pixels OFF would be included). These higher frequencies were thought to be able to provide greater rotation discrimination. On the other hand, using 128X128 CCD pixels could put greater emphasis on the lower frequencies that usually determine overall size. The simplest correlation, a "+" with an "X" was a rotation problem, so since scale would be more difficult, the 128X128 region was used.

A second, less complicated, but important, issue here was the positioning of the camera such that the Fourier transform was exactly centered on the 128X128 pixel area captured during the framegrab. Software designed to aid in this positioning is described in Appendix G.

5.2.4 Binarize $|FT|^2$. The next decision related to step 4: How should the $|FT|^2$ features be binarized?

This time the nonuniformity of response of the camera and distortion through the optical system would apparently not be reduced by the subtraction of two frames as was the case in the JTC. Not so! By displaying a blank MOSLM and capturing its resultant frame at CCD1, an estimate of the system "bias" was acquired. By subtracting this frame from the ones taken for the scene and template, an improved $|FT|^2$ feature set was provided.

With that problem solved, the next job was to decide on a threshold for binarization. Through subjective empirical analysis, the threshold decided on was five times the mean value of the entire Fourier plane (after subtraction of the bias frame).

5.2.5 Display $|FT|^2$ on SLM. The first consideration in step 5 was that the binarized $|FT|^2$ had to fit within the MOSLMs 128X128 format. Since only a 128X128 pixel region was captured in step 3, this was not a problem. However, it may become important for future research.

A second issue, that was not remedied, was the fact that the TARGA framegrabber system had a pixel aspect ratio of .97 to 1 instead of the 1 to 1 ratio of the MOSLM. Although this was not a problem for the JTC of Chapter III, it would cause problems in performing the LPCT.

5.2.6 Capture LPCT-$|FT|^2$. After displaying the $|FT|^2$ patterns on the MOSLM, the LPCT had to be captured. The horizontal extent of the captured area was fixed, since this dimension related to the rotation angle, and a full $2\pi$ had to be included in the captured region. (Reference to Figure 24 of Chapter IV may be useful here.)

However, the concerns of Kobel and Martin mentioned in section 5.2.3 appeared again in this step, and a decision about the vertical extent to be included had to be made. In fact, a third variable, the magnification provided by the two lens (L2 and L3) imaging system could be changed (within limits) to vary the vertical spread of spatial frequencies in this LPCT-$|FT|^2$ space. The mirror images, as well as the Fourier transform artifacts should be avoided too. Not to mention that the effect of the two bad lines of the MOSLM must be minimized.

All of these factors, and the fact that the Perkin-Elmer sample was only on loan for three weeks, resulted in the following decision: The area captured was 110X456 CCD2 pixels, which included radii of between 15 and 60 MOSLM pixels or 30 and 120 CCD1 pixels in the $|FT|^2$ space. These related to spatial frequencies of between 3 and 12 MOSLM pixel period in the original scene and template space.

5.2.7 Binarize LPCT-$|FT|^2$. Childress and Walrond expressed concern about the spread of the spatial frequencies along the vertical direction [9:97]. This spread relates to a resolution of scale used for template rectification. To minimize this problem the JTC used for correlation of the LPCT-$|FT|^2$ features would use inputs that were 55X114 pixels, instead of the 45X128 pixel inputs of the JTC used for the FLIR inputs of Chapter III. Additional gains were made since the original grabbed frame was 110X456 pixels, so a 4 for 1 reduction in the rotation axis was made while only a 2 for 1 reduction was made along the scale axis.

A system bias blank MOSLM frame was captured like the one described in step 3 above, and it was subtracted from the LPCT-$|FT|^2$ feature grabbed frames before binarization. Binarization was performed using the mean value of the entire grabbed region minus the bias frame. This was arrived at by subjective empirical means.

5.2.8 Correlate LPCT-$|FT|^2$. Now that a binarized pair of LPCT-$|FT|^2$ features were available for display on the MOSLM, correlation could be performed. Using this 55X128 pixel input configuration meant that only a 128X128 pixel area in the joint transform plane could be captured. If a 256X256 pixel region were captured instead, overlap of various orders in the correlation plane might occur. This overlap could result in incorrect correlation peak identification.

Various versions of templates were tried in this correlator, including the normal one, a 180°-only version, and a negative of the 180°-only version. The 180°-only version was tried because of Childress and Walrond's concern for rotation ambiguity [9:96]. The rationale for the 180° template was that a $1\pi$ vs $2\pi$ correlation would perform the same function as the $2\pi$ vs $4\pi$ correlation that they suggested. In either case, a 180° ambiguity would still exist due to the symmetry of the $|FT|^2$.

One final note here is that the energy typically displayed in these LPCT-$|FT|^2$ JTC input planes is greater than those of the input planes of the JTC of Chapter III. For that reason, additional neutral density filtering was necessary between L1 and CCD1.

## 5.3 Experimental Results

Only test patterns were used to correlate in the LPCT-$|FT|^2$ feature space. The patterns selected were a "+" and an "x" to test rotation invariance, and squares of different sizes to test scale invariance.

All of the results have the same layout, their order of display was governed by the limits of the framegrabber system. The figures should be read as a book, from left to right in four columns, then top to bottom in three rows. In the first row are: a) optically generated $|FT|^2$ of the scene; b) a minus scene bias frame, binarized on 5 times the mean difference; c) optically generated $|FT|^2$ of the template; d) c minus scene b.as frame, binarized on 5 times the mean difference. In the second row are: e) original inputs with scene on top and template on bottom; f) optically generated LPCT of b and d (increasing ln(r) is downward on page); g) input to the JTC which is the two LPCTs of f minus a system bias frame, binarized on the mean difference of each. In the third row are: h) optically generated $|FT|^2$ of g; i) optically generated $|FT|^2$ of top half of g, which are the features of the scene; j) h minus i, binarized on the mean difference; k) optically generated $|FT|^2$ of j, which is the correlation plane.

Figure 30 shows the system results for a "+" and an "x" using the fall 360° version of the template. Figure 33a represents a side view of a surface plot of the correlation plane in Figure 30k. Only the top crosscorrelation region is represented, and the view is along the vertical direction. Notice the peak on the right hand side of 33a, which corresponds to one of the dark spots in the upper right hand portion of Figure 30k. Both the peak and the spot are roughly in the location of a 45° rotation. Secondary peaks also exist at locations of about -45° and +/-135°.

Figures 31 and 32 are the system results using a 180°-only template and a negative of the 180°-only template. Figures 33b and 33c are to Figures 31k and 32k respectively, as Figure 33a is to Figure 30k. No benefit was apparently derived from these attempts at modifying the template.

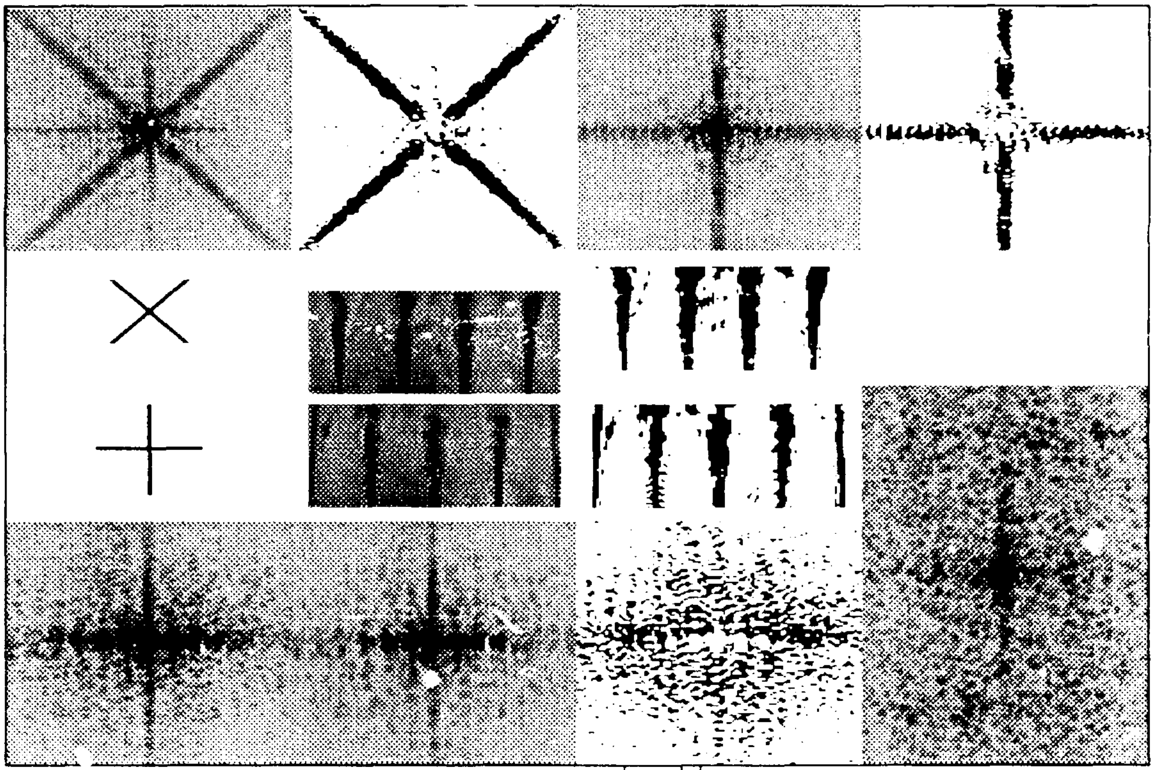

**Figure 30.** Correlation of LPCT-$|FT|^2$ features of an "x" and a "+" using the hybrid architecture.

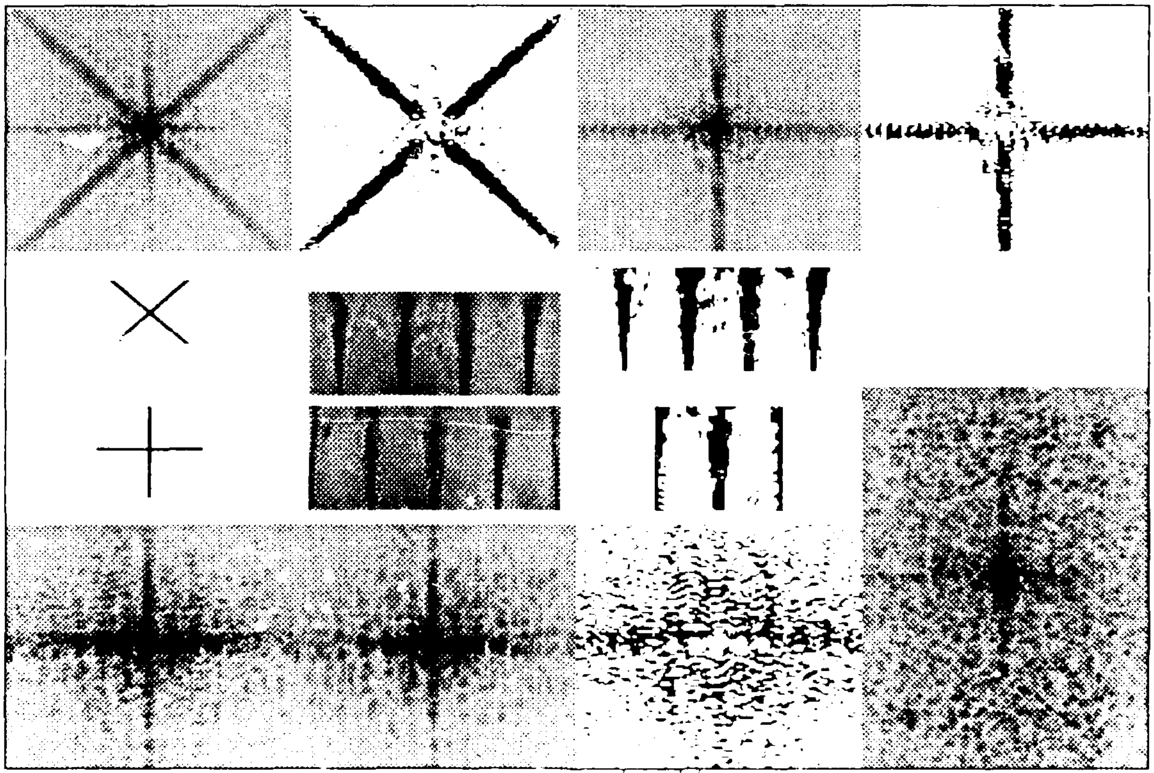

**Figure 31.** Correlation of LPCT-$|FT|^2$ features of "x" and "+" using hybrid architecture with 180°-only template.

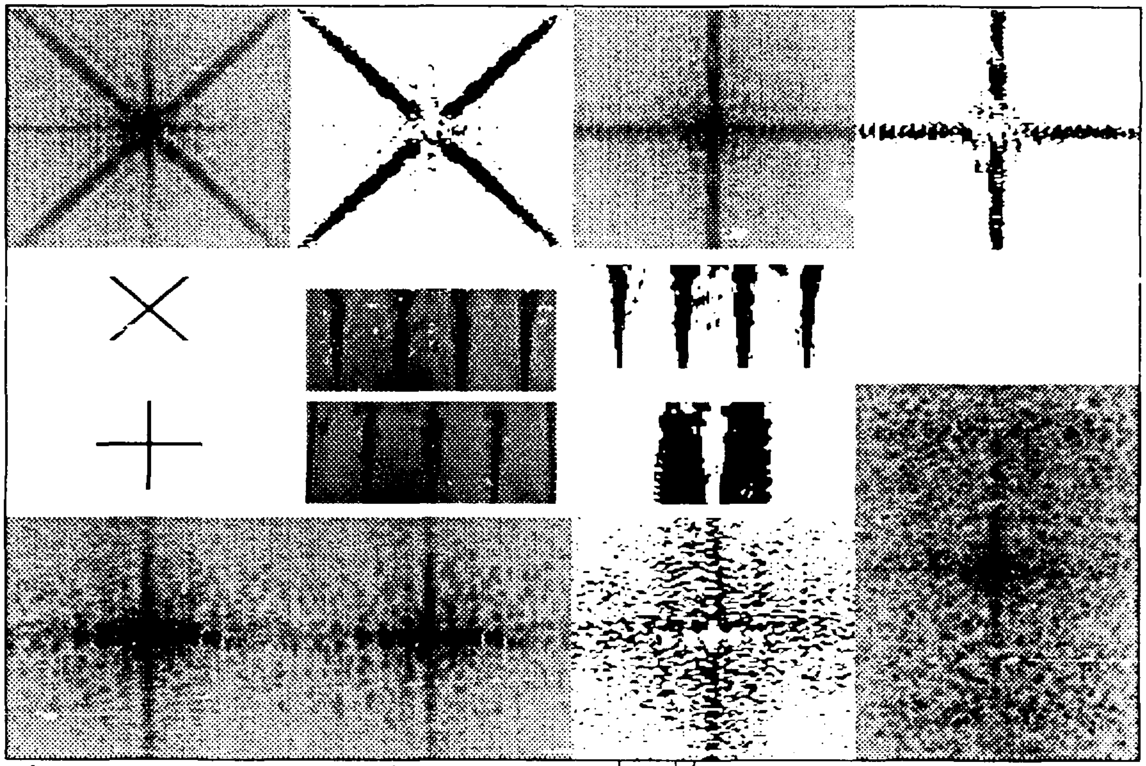

**Figure 32.** Correlation of LPCT-$|FT|^2$ features of "x" and "+" using the h;brid architecture. Negative 180° template used.

Figures 34 and 35 are the system results of correlating the LPCT-$|FT|^2$ features of squares fo different sizes. In Figure 34, the scene square is larger than the template, and in Figure 35, it is smaller. Figures 36a and 36b, which are horizontal sideviews of the surface plots of 34k and 35k respectively, demonstrate the extremely noisy correlation plane results. All along the vertical axis, high energy locations that may be misinterpreted as correlation peaks exist. This is caused by the pixelization of the fringe pattern of the joint transform pl ne, and its effect can only be reduced by increasing the number of display pixels available on the MOSLM (say from 128X128 to 1024X1024).

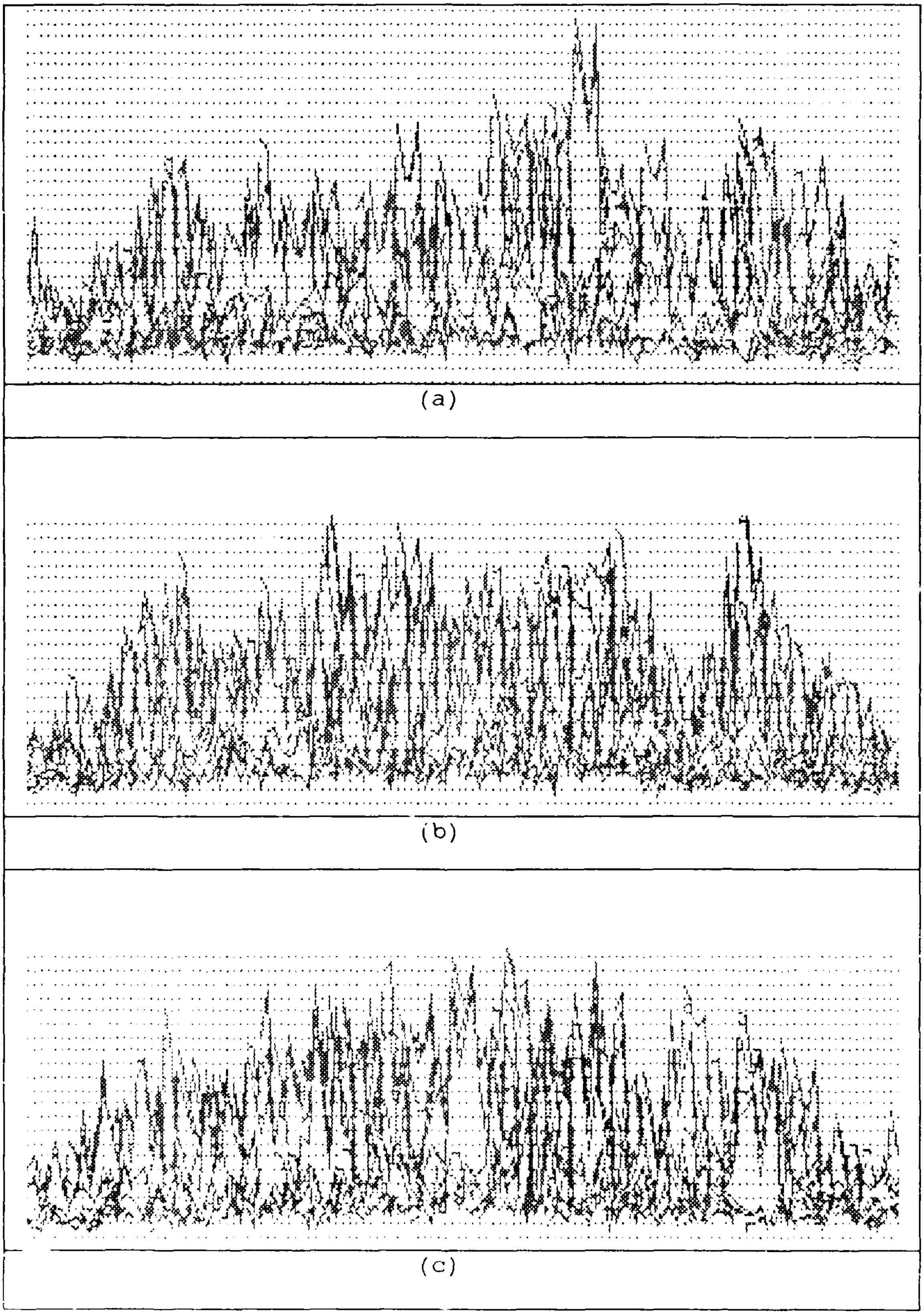

**Figure 33.** Vertical sideviews of top portion of correlation planes in: a) Figure 30k; b) Figure 31k; and c) Figure 32k.

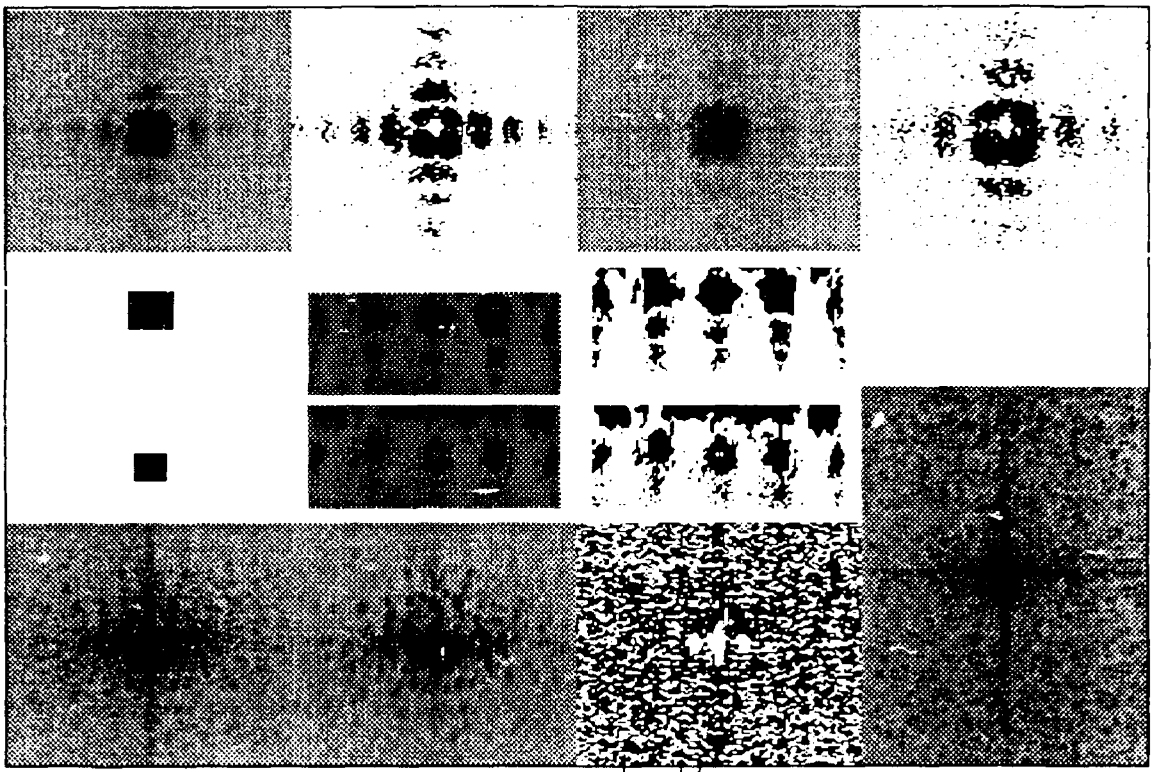

**Figure 34.** Correlation of LPCT-|FT|$^2$ features of large square with smaller template using hybrid architecture.

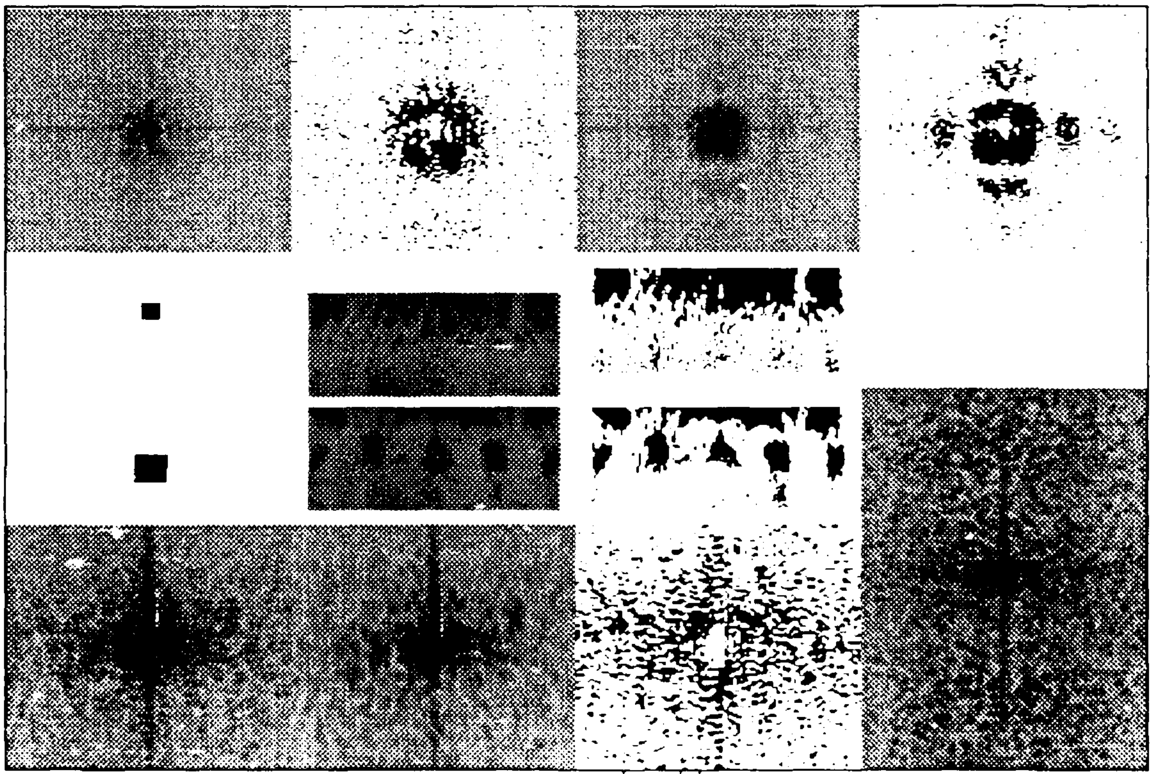

**Figure 35.** Correlation of LPCT-|FT|$^2$ features of a small square with a larger template using hybrid architecture.

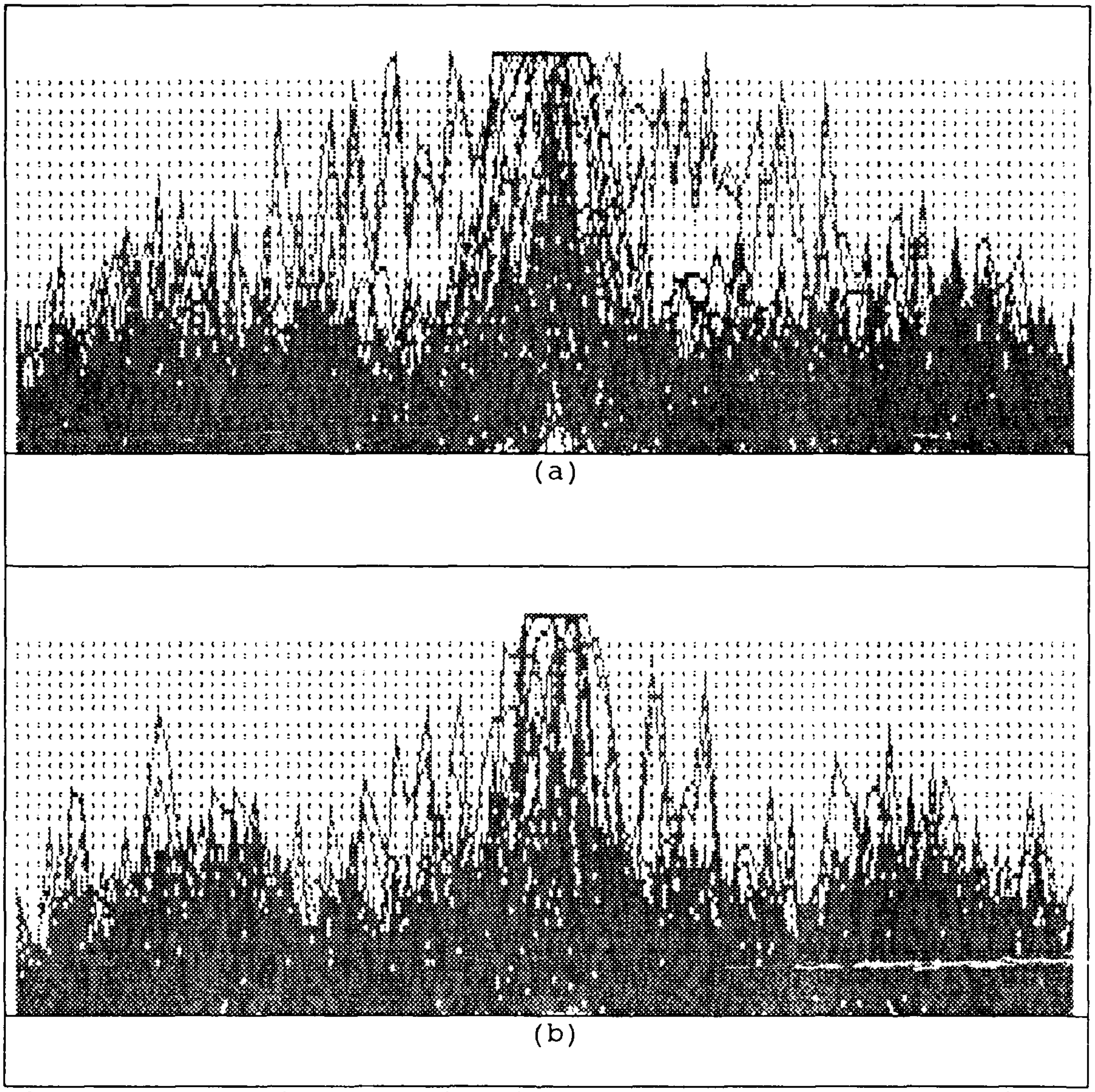

**Figure 36.** Horizontal sideviews of correlation planes in: a) Figure 34k; and b) Figure 35k.

This fact was contributing to the on axis false alarms of chapter III.

Note also the difficulty of acquiring a good binary $|FT|^2$ feature set in Figure 35b. The small number of pixels illuminated for the small scene square of 35c deliver inadequate energy to the $|FT|^2$ features of 35a to be properly binarized. If the camera had greater dynamic range, or if the neutral density filtering could be controlled by the amount of energy input (i.e. the number of ON pixels at the MOSLM) this problem might be reduced. Other sources of distortion, such as the nonlinear response of the CGH or the effects of binarization at the various stages may also be contributing to a very noisy correlation plane. All of these variables are not understood well enough to make proper comment here.

## 5.4 Summary

This chapter discussed the hybrid optical/digital architecture designed and implemented for this thesis. This architecture was to provide a position, scale, and rotation invariant pattern recognition capability. The chapter described the architecture, discussed engineering considerations related to its design, and presented experimental results.

Only marginal results were achieved using this architecture, as evidenced by the very noisy correlation planes. The effects of the binarization at various stepsis not well understood, and may be one source of difficulty in achieving proper correlation in this LPCT-$|FT|^2$ feature space.

In light of the poor correlation results in the LPCT-$|FT|^2$ feature space, no attempts at template rectification, and subsequent correlation in the original scene space were made.

# VI. CONCLUSION

This chapter presents a summary of thesis results, provides conclusions based upon these results, and offers recommendations for future research effort.

## 6.1 Summary

There were three major goals of this thesis: 1) To develop software for PC control of display of real FLIR imagery on the MOSLM, and the subsequent capture of frames on the CCD cameras; 2) To implement and test an optical joint transform correlator using the techniques and data of the first goal; and 3) To design, implement, and test a hybrid optical/digital architecture for distortion invariant pattern recognition like the KMH algorithm, employing both the joint transform correlator, and a CGH to perform the LPCT.

Two binarization techniques were discovered during this thesis effort. The first was developed for binarization of input FLIR imagery. This technique, described in section 3.2.3, provided improved correlation results, shown in section 3.3, over those obtained for FLIR imagery that had not been preprocessed. The key to this technique was that only pixels that were both brighter than the scene average, and brighter than a local 3 pixel by 3 pixel average were illuminated.

The second binarization technique involved binarization in the joint transform plane of the JTC. This technique, described in section 3.2.4, provided improved correlation results, shown in section 3.2.4 and 3.2.5, over other published binary JTC techniques. The key to this technique was the subtraction of the $|FT|^2$ of the scene alone from the $|FT|^2$ of the template and scene displayed together.

The performance of an interferogram LPCT CGH created by Mayo [8] and Hill [12] was compared with that of a CGH created by E-beam lithography on loan to AFIT from Perkin-Elmer Corporation [22]. The Perkin-Elmer CGH provided superior performance both in diffraction efficiency and image clarity (see Chapter IV). However, due to time constraints, only subjective measures were possible.

The hybrid architecture was used to perform correlation of the LPCT-$|FT|^2$ features of geometric objects. Only marginal results, discussed in section 5.3, were obtained. For that reason attempts at performing template rectification, and subsequent correlation in the original image space were not made as discussed in section 5.1.

## 6.2 Conclusions

The binarization techniques employed for both the FLIR input data, and the fringes in the joint transform plane proved successful, and show great promise as improvements on existing correlation techniques.

The JTC implemented for this thesis is a strong candidate for use in a tracking system (after target acquisition). It could also be used in more controlled environments, such as assembly line inspection, where rotation and scale invariance may not be so important.

The improved performance of the Perkin-Elmer E-beam sample over the Mayo-Hill interferogram is not conclusive in itself, although subjective testing in this thesis does provide strong evidence. Only proper modelling of the two samples, as well as the limits of both techniques can shed sufficient light on this subject.

In light of the marginal results achieved with the above architecture on even simple geometric objects, much analysis is necessary. Several speculations are offered, that only proper modelling and additional testing can verify. The additive effects of binarization and pixelization at the various stages may have corrupted the LPCT-$|FT|^2$ beyond the recognition capability of the JTC.

## 6.3 Recommendations for Future Research

Recommendations for follow on efforts follow two tracks, one along the lines of PSRI pattern recognition, another along the lines of JTC implementation. A list of possible research areas is provided.

1. Investigation of alternate frame capture regions, template creation and binarization techniques are recommended. These may have prohibited success in correlating the LPCT-$|FT|^2$ features.

2. The KMH algorithm should be retested digitally using

some realistic data set. Possible candidates could include FLIR or SAR data. These data may first be segmented using any number of segmentation techniques, including the Gabor transform [24], or techniques employed by Mike Roggemann [25]. The digital implementation may provide important insights into the optical implementation and should be made to model as many limitations of the optical system as possible.

3. Proper mathematical modelling of this architecture, as well as any other future implementation, will help identify architectural weaknesses. These include weaknesses both in hardware, such as insufficient input SLM resolution or camera resolution, or software, such as area of capture and binarization techniques.

4. If proper modelling proves the E-beam technique to be superior, an in house capability of developing E-beam CGHs could be developed with the help of WRDC/EL who possess the necessary equipment.

5. LCTVs should be investigated as a substitute for the MOSLMs used in this architecture to eliminate some of the binarization effects.

6. The CCD cameras should be considered for replacement with either a CCD of greater resolution or a CID

framegrabber system.

7. Other techniques for developing the LPCT should be considered, including fiber bundles, unique focal plane array mappings (wedge ring etc.) and digital mapping.

8. The necessary code for template rectification should be written.

9. Techniques for optically implementing the phase reinsertion step of the KMH algorithm should be developed.

10. An alternative architecture for the JTC, that would reduce its length from two focal lengths to one focal length, should be investigated [1:84]. It is pictured in Figure 37.

11. Finally, a JTC fringe binarization threshold based on the average difference could be sped up by measuring the total energy in the $|FT|^2$ with a separate detector off a sampling beam splitter. Presently, determination of this threshold requires summing the pixel values accross the array (See Figure 37).

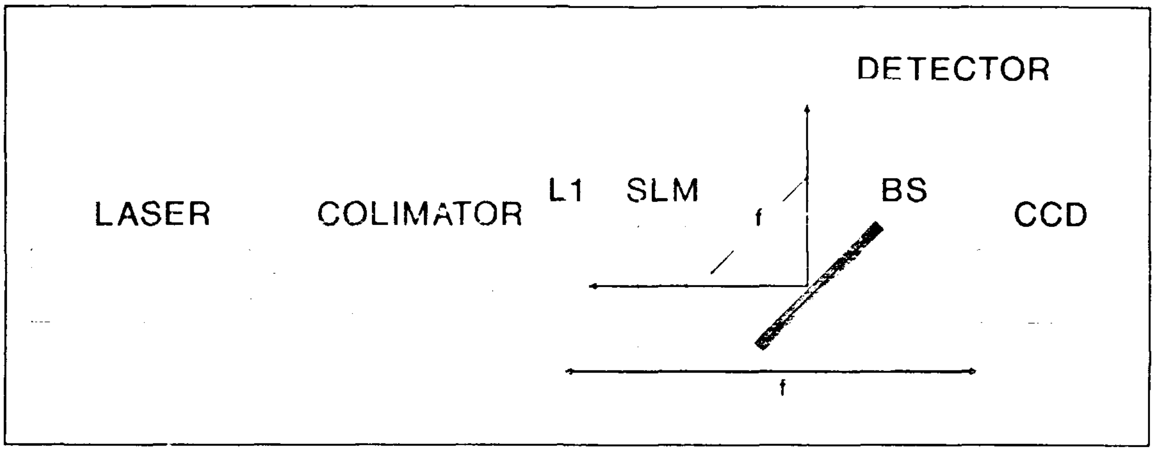

**Figure 37.** Possible improvement on joint transform correlator, featuring shorter total length (1f vs normal 2f), and a detector to measure total energy for faster fringe binarization calculations.

# APPENDIX A. MAGNETO-OPTIC SPATIAL LIGHT MODIULATOR (MOSLM)

This appendix presents information relating to the MOSLM used in the experiments of this thesis. First a brief description of the operation of the MOSLM is given, then some useful programs will be described. Computer code written during this thesis effort in "C" for Borland's Turbo-C compiler is also included.

## A.1 MOSLM Operation

The following discussion should serve only as a supplement to the Semetex manual [26]. Technical representatives at Semetex can prcvide additional information as well [27].

Although technically incorrect, the MOSLM can be thought of as an array of 128X128 rectangular Pockels cells. The operation of the individual pixels differ from a Pockels cell, in that the MOSLM uses the magneto-optic effects of the iron garnet material it is made up of, rather than the electro-optic effect that Pockels cells employ.

The MOSLM modifies the index of refraction of the iron garnet material by applying a magnetic field across a pixel. This field is applied through two wires, aligned along the cardinal axes of the array, that cross at one corner of the pixel of interest. After all the selected pixels in the array are individually "nucleated", a write pulse is applied to the entire array. The effect here is to increase the contrast of the image created by the earlier nucleation. Once a pixel has been nucleated, it can only be changed by erasing the entire array with an erase pulse.

A second use of the MOSLM not investigated during this thesis effort is that of a binary phase only filter. Since the nucleation of the pixel changes the optical path length of light rays passing through it, the entire array can provide binary phase only modulation of the incident wavefront by removal of the analyzer (output polarizer). In either phase modulation or the above described amplitude modulation operation, the input polarizer may not be necessary, since its only purpose is to properly polarize an already polarized collimated laser beam.

The individual pixels are addressed through an interface card located in the PC. The card occupies segment hex B000 in the software listed later in these appendices, however dip switches on the card allow for this base address to be changed. Each byte of data written to an offset address in segment hex B000 represents eight pixels. The offset addresses range from hex 0000 to 07FF and increase from left to right then top to bottom accross the array. The eight pixels of each address are in the same column but in eight successive rows. The least significant bit of each byte of data represents the top pixel and the most significant the bottom pixel of the eight pixel byte. Writing a zero to the proper bit will nucleate the pixel, while writing a 1 will leave it unchanged. To create erase and write pulses data must be sent to offset addresses hex 0800 and 0801 respectively.

Although a maximum of eight pixels could be written to at a time with this digital circuitry, testing showed that any more than two pixels would result in inadequate performance in either of two devices. This is reflected in all of the software in these appendices.

Proper alignment of the MOSLM requires adjustment of five potentiometers in the MOSLM. Improper alignment may result in pixels being nucleated before being written to (leading pixels) or pixels not being nucleated after being written to. The Semetex manual suggests a software generated test signal be applied to the device during alignment, but this software was not provided. The next section describes this software as well as some other useful programs for use with the MOSLM.

## A.2 MOSLM Software

In addition to the software developed for this thesis effort and presented below, there is also additional software available for the control of the MOSLM. The program SMDISP.C and associated files can be found in the distribution diskettes provided by Semetex. This program allows for the display of files formatted as described in the manual.

Four programs are presented here that required no control of the CCD camera to be useful. The first is

SLM_TEST.C the program for alignment of the MOSLM. It is assumed that a camera will be operated in live mode and placed in an image plane to monitor the MOSLM display during execution. The next two are CHECKER.C and CHECKER2.C that display checkerboard patterns on the MOSLM. With a camera in the Fourier plane in live mode, these two programs can lend insight into appropriate capture regions for the framgrabber. The difference between the two is that pattern of the first is one pixel on, one pixel off, and the second is two on, two off. The last program, X-HAIR.C displays a large crosshair on the MOSLM, that can be used for optical system alignment.

Additional software controlling the MOSLM will also appear in Appendix G. To understand that code it is also necessary to understand the operation of the TARGA framegrabber system discussed in Appendix B.

```
/*************************************************************************/
/* SLM_TEST.C                                                            */
/*                                                                       */
/*                 AIR FORCE INSTITUTE OF TECHNOLOGY                     */
/*                     PROGRAM FOR SLM ALIGNMENT                         */
/*                        by Capt John Cline                             */
/*                          June 29, 1988                                */
/*                                                                       */
/*This program is a variation of the alignment routine suggested by     */
/*the Semetex manual.  I believe it does a better job of identifying    */
/*leading pixels.                                                        */
/*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "time.h"
unsigned r, q;

main()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;          /*SLM is at b000 segment*/
   *(semetex+0x800)=1;                                       /*erase SLM*/
   wait(1000);                                     /*wait or lose pixels*/

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
   printf("writing horizontal lines \n");
   do
   {
      for(q=0x0; q<0x800; q=q+0x80)
      {
         for(r=0x0; r<0x40; r++)                /*choose only left half*/
         {
            *(semetex+q+r)=0xfe;            /*top row of eight row byte*/
            wait(5);                       /*slow down so you can watch*/
         }
         for(r=0x0; r<0x40; r++)
         {
            *(semetex+q+r)=0xfb;                            /*third row*/
            wait(5);
         }
         for(r=0x0;r<0x40;r++)
         {
            *(semetex+q+r)=0xef;                            /*fifth row*/
            wait(5);
         }
         for(r=0x0;r<0x40;r++)
         {
            *(semetex+q+r)=0xbf;                          /*seventh row*/
            wait(5);
         }
      }
```

```
for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x0; r<0x40;r++)
   {
      *(semetex+q+r)=0xfd;                          /*second row*/
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xf7;                          /*fourth row*/
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xdf;                           /*sixth row*/
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0x7f;                          /*eighth row*/
      wait(5);
   }
}

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x40; r<0x80;r++)              /*choose only right half*/
   {
      *(semetex+q+r)=0xfe;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xfb;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xef;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xbf;
      wait(5);
   }
}

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x40; r<0x80;r++)
   {
      *(semetex+q+r)=0xfd;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xf7;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xdf;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0x7f;
      wait(5);
   }
}

wait(3000);                                 /*take a quick look*/
*(semetex+0x801)=1;       /*pulse the SLM for improved contrast*/
wait(5000);                                 /*take another look*/
*(semetex+0x800)=0;                             /*erase the SLM*/

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x40; r<0x80;r++)    /*start with right side first now*/
   {
      *(semetex+q+r)=0xfe;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xfb;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xef;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xbf;
      wait(5);
   }
}

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x40; r<0x80;r++)
   {
      *(semetex+q+r)=0xfd;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xf7;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0xdf;
      wait(5);
   }
   for(r=0x40;r<0x80;r++)
   {
      *(semetex+q+r)=0x7f;
      wait(5);
   }
}

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x0; r<0x40;r++)                /*then the left side*/
   {
      *(semetex+q+r)=0xfe;
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xfb;
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xef;
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xbf;
      wait(5);
   }
}

for(q=0x0; q<0x800; q=q+0x80)
{
   for(r=0x0; r<0x40;r++)
   {
      *(semetex+q+r)=0xfd;
      wait(5);
   }
   for(r=0x0;r<0x40;r++)
   {
      *(semetex+q+r)=0xf7;
```

```
                wait(5);
            }
            for(r=0x0;r<0x40;r++)
            {
                *(semetex+q+r)=0xdf;
                wait(5);
            }
            for(r=0x0;r<0x40;r++)
            {
                *(semetex+q+r)=0x7f;
                wait(5);
            }
        }

        wait(3000);
        *(semetex+0x801)=1;                                 /*contrast*/
        wait(5000);
        *(semetex+0x800)=0;                                    /*erase*/
        printf("type q if you want to go on to vertical \n");
    }
    while(getch()!='q');

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
    printf("writing vertical lines \n");
    do
    {
        for(r=0x00; r<0x80; r=r+0x02) /*even columns starting with zero*/
        {
            for(q=0x000; q<0x400; q=q+0x080)            /*top half only*/
            {
                *(semetex+q+r)=0x3f;          /*write two bytes at a time*/
                *(semetex+q+r)=0xcf;
                *(semetex+q+r)=0xf3;
                *(semetex+q+r)=0xfc;
                wait(20);                      /*slow down so you can see*/
            }
        }
        for(r=0x01; r<0x80; r=r+0x02)   /*odd columns starting with one*/
        {
            for(q=0x000; q<0x400; q=q+0x80)
            {
                *(semetex+q+r)=0x3f;
                *(semetex+q+r)=0xcf;
                *(semetex+q+r)=0xf3;
                *(semetex+q+r)=0xfc;
                wait(20);
            }
        }
        for(r=0x00; r<0x80; r=r+0x02)
        {
            for(q=0x400; q<0x800; q=q+0x080)         /*bottom half only*/
            {
                *(semetex+q+r)=0x3f;
```

```
      *(semetex+q+r)=0xcf;
      *(semetex+q+r)=0xf3;
      *(semetex+q+r)=0xfc;
      wait(20);
   }
}
for(r=0x01; r<0x80; r=r+0x02)
{
   for(q=0x400; q<0x800; q=q+0x80)
   {
      *(semetex+q+r)=0x3f;
      *(semetex+q+r)=0xcf;
      *(semetex+q+r)=0xf3;
      *(semetex+q+r)=0xfc;
      wait(20);
   }
}
wait(3000);
*(semetex+0x801)=1;                                  /*contrast*/
wait(5000);
*(semetex+0x800)=0;                                     /*erase*/

for(r=0x00; r<0x80; r=r+0x02)
{
   for(q=0x400; q<0x800; q=q+0x080)     /*now start with bottom*/
   {
      *(semetex+q+r)=0x3f;
      *(semetex+q+r)=0xcf;
      *(semetex+q+r)=0xf3;
      *(semetex+q+r)=0xfc;
      wait(20);
   }
}
for(r=0x01; r<0x80; r=r+0x02)
{
   for(q=0x400; q<0x800; q=q+0x80)
   {
      *(semetex+q+r)=0x3f;
      *(semetex+q+r)=0xcf;
      *(semetex+q+r)=0xf3;
      *(semetex+q+r)=0xfc;
      wait(20);
   }
}
for(r=0x00; r<0x80; r=r+0x02)
{
   for(q=0x000; q<0x400; q=q+0x080)          /*then do top half*/
   {
      *(semetex+q+r)=0x3f;
      *(semetex+q+r)=0xcf;
      *(semetex+q+r)=0xf3;
      *(semetex+q+r)=0xfc;
      wait(20);
```

```
         }
      }
      for(r=0x01; r<0x80; r=r+0x02)
      {
         for(q=0x000; q<0x400; q=q+0x80)
         {
            *(semetex+q+r)=0x3f;
            *(semetex+q+r)=0xcf;
            *(semetex+q+r)=0xf3;
            *(semetex+q+r)=0xfc;
            wait(20);
         }
      }
      wait(3000);
      *(semetex+0x801)=1;                                  /*contrast*/
      wait(5000);
      *(semetex+0x800)=0;                                     /*erase*/
      printf("type q if you want to quit \n");
   }
   while(getch()!='q');
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
/*TURBO-C has a time delay function "delay()" that could be used    */
/*instead of this wait() subroutine.                                */

wait(t)
{
   time_t start, finish;
   time(&start);
   time(&finish);
   while(difftime(finish, start) < t/1000)
   {
      time(&finish);
   }
}
/*********************************************************************/
```

```
/***********************************************************************/
/* CHECKER.C                                                           */
/*                                                                     */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/*              CHECKERBOARD PATTERN GENERATOR FOR THE SLM             */
/*                        by Capt John Cline                           */
/*                          August 31, 1989                            */
/*                                                                     */
/*This program will display a checkerboard pattern on the SLM.  This */
/*is useful for bench alignment, since the maximum spatial frequency */
/*will be present in the Fourier plane.  Another program, CHECKER2.C,*/
/*will display a checkerboard pattern of half the maximum spatial    */
/*frequency, or 2 pixels ON, 2 pixels OFF.                           */
/***********************************************************************/

#include "stdio.h"
#include "string.h"
#include "time.h"
unsigned r, q;

main()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;          /*SLM is at b000 segment*/
   *(semetex+0x800)=1;                                      /*erase SLM*/
   wait(1000);                                     /*wait or lose pixels*/

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
   for(q=0x000; q<0x800; q=q+0x080)
   {
      for(r=0x00; r<0x80; r=r+2)
      {
         *(semetex+q+r)=0xfe;
         *(semetex+q+r)=0xfb;
         *(semetex+q+r)=0xef;
         *(semetex+q+r)=0xbf;
      }
      for(r=0x01; r<0x80; r=r+2)
      {
         *(semetex+q+r)=0xfd;
         *(semetex+q+r)=0xf7;
         *(semetex+q+r)=0xdf;
         *(semetex+q+r)=0x7f;
      }
   }
   *(semetex+0x801)=0;
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
/*TURBO-C has a time delay function "delay()" that could be used     */
/*instead of this wait() subroutine.                                 */

wait(t)
```

```
{
   time_t start, finish;
   time(&start);
   time(&finish);
   while(difftime(finish, start) < t/1000)
   {
      time(&finish);
   }
}
/********************************************************************/
```

```
/**************************************************************************/
/* CHECKER2.C                                                             */
/*                                                                        */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                       */
/*            CHECKERBOARD PATTERN GENERATOR FOR THE SLM                  */
/*                      by Capt John Cline                                */
/*                       August 31, 1989                                  */
/*                                                                        */
/*This program will display a checkerboard pattern on the SLM.            */
/*Unlike CHECKER.C, which turns one pixel ON, then one pixel OFF,         */
/*this program turns two pixels ON, then two pixels OFF.                  */
/**************************************************************************/

#include "stdio.h"
#include "string.h"
#include "time.h"
unsigned r, q;

main()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;          /*SLM is at b000 segment*/
   *(semetex+0x800)=1;                                     /*erase SLM*/
   wait(1000);                                     /*wait or lose pixels*/

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
   for(q=0x000; q<0x800; q=q+0x080)
   {
      for(r=0x00; r<0x80; r=r+4)
      {
         *(semetex+q+r)=0xfc;
         *(semetex+q+r)=0xcf;
         *(semetex+q+r+1)=0xfc;
         *(semetex+q+r+1)=0xcf;
      }
      for(r=0x02; r<0x80; r=r+4)
      {
      *(semetex+q+r)=0xf3;
         *(semetex+q+r)=0x3f;
         *(semetex+q+r+1)=0xf3;
         *(semetex+q+r+1)-0x3f;
      }
   }
   *(semetex+0x801)=0;
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
/*TURBO-C has a time delay function "delay()" that could be used      */
/*instead of this wait() subroutine.                                  */

wait(t)
{
   time_t start, finish;
   time(&start);
   time(&finish);
   while(difftime(finish, start) < t/1000)
   {
      time(&finish);
   }
}
/*********************************************************************/
```

```
/**************************************************************************/
/* X_HAIR.C                                                               */
/*                                                                        */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                       */
/*             CROSSHAIR PATTERN GENERATOR FOR THE SLM                    */
/*                        by Capt John Cline                              */
/*                          August 31, 1989                               */
/*                                                                        */
/*This program will display a crosshair pattern on the SLM.  This is */
/*useful for bench alignment, since the diffraction pattern in any   */
/*plane (image, Fourier, or anywhere in between) resembles a         */
/*crosshair.  As successive optical elements are inserted into the   */
/*optical path, the optic axis should not move.  Therefore, even     */
/*though the pattern that the crosshair makes, as viewed by a fixed  */
/*camera, will change with addition of optical elements, the spot    */
/*where the center of the crosshair like pattern is located should   */
/*remain the same provided the SLM centers the original crosshair on */
/*the optic axis.                                                    */
/**************************************************************************/

#include "stdio.h"
#include "string.h"
#include "time.h"
unsigned r, q;

main()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;         /*SLM is at b000 segment*/
   *(semetex+0x800)=1;                                    /*erase SLM*/
   wait(1000);                                  /*wait or lose pixels*/

   /*write four columns*/
   for(q=0x0; q<0x800; q=q+0x80)
   {
      *(semetex+q+63)=0xfc;
      *(semetex+q+63)=0xf3;
      *(semetex+q+63)=0xcf;
      *(semetex+q+63)=0x3f;
      *(semetex+q+64)=0xfc;
      *(semetex+q+64)=0xf3;
      *(semetex+q+64)=0xcf;
      *(semetex+q+64)=0x3f;
      *(semetex+q+62)=0xfc;
      *(semetex+q+62)=0xf3;
      *(semetex+q+62)=0xcf;
      *(semetex+q+62)=0x3f;
      *(semetex+q+65)=0xfc;
      *(semetex+q+65)=0xf3;
      *(semetex+q+65)=0xcf;
      *(semetex+q+65)=0x3f;
   }
   /*write four rows*/
   for(q=0x0; q<0x80; q++)
   {
      *(semetex+q+128*7)=0x3f;
      *(semetex+q+128*8)=0xfc;
   }
   *(semetex+0x801)=0;                    /*write pulse for contrast*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
/*TURBO-C has a time delay function "delay()" that could be used    */
/*instead of this wait() subroutine.                                 */

wait(t)
{
   time_t start, finish;
   time(&start);
   time(&finish);
   while(difftime(finish, start) < t/1000)
   {
      time(&finish);
   }
}
/*********************************************************************/
```

# APPENDIX B. TARGA FRAMEGRABBER SYSTEM

This appendix presents information relating to the AT&T Truevision Advanced Raster Graphics Adapter (TARGA) framegrabber board for IBM compatible PCs used in the experiments of this thesis. First a brief description of the operation of the TARGA is given, including its file format. Next some software provided by the vendor will be described. Finally, batch and command files (.CMD) for use with the vendor program TESTTARG.EXE will be presented.

The following discussion should serve only as a supplement to the TARGA manuals [28] [29] and documentation files provided with the vendor software [30] [31] [32] [33]. Information concerning the SONY CCD cameras used with the TARGA system can be found in its manual [34].

## B.1 TARGA Operation

The TARGA framegrabber is capable of taking a video signal input and digitizing it. It can accept NTSC video from any source including a VCR or camera. The digitized signal is 400 pixel lines by 512 pixel columns and samples about 80% of the normal monitor display in both dimensions. The individual pixels are represented by eight bit grey scale using unsigned byte format. The brightest pixels will have a value of hex FF and the darkest hex 00. Although other Truevision hardware products allow for color images, the TARGA 8 boards available at AFIT are only black and white devices.

Not only can these T^RGA boards digitize video inputs, but they can also store these digitized images in a file. These files can be recalled to the board later for redisplay on a video monitor (not the PC monitor) using the VIDEO OUT port of the card. The VIDEO OUT port of the card will either transmit the signal of the VIDEO IN port or the digitized image resident on the card depending on software selection of these two alternatives.

Files saved by the TARGA 8 boards all have the same format. Unlike most other raster image files whose origins are in the upper left hand corner, the origin of TARGA 8 files created by the vendor software is in the lower left hand corner. After 18 bytes of header data, the grey scale pixel data is listed serially from left to right starting from the bottom row. The actual header format for the TARGA 8 data type (type 3) is not listed in the manuals provided by the vendor, but it appeared as though it was the same as data type 1 that was detailed in the TARGA manual [28:E-2].

Dip switches on the TARGA board allowed the user to define several important specifications for addressing the TARGA board. Switches could select both a segment and an offset address for the board [28:B-1]. The segment and offset used throughout this thesis were hex D000 and 0220 respectively.

The DOS environmental variable TARGA must be set to reflect these before running software that uses the TARGA board. The variable TARGASET must also be initialized to board. The variable TARGASET must also be initialized to specify capture region, board type, and sync source.

## B.2 TARGA Software

Software for the TARGA can be divided into four categories: 1) TRUEART; 2) C library routines; 3) utilities; and 4) batch files and TESTTARG command (.CMD) files.

The program TRUEART.EXE was supplied with the TARGA 8 boards when they were purchased [30], and offers user friendly control of the TARGA 8 board. Documentation for this program is included in the manual [28:3-1].

The C library routines are a collection of programs written for Microsoft-C compilers, including Quick-C. They were not used during this or any other thesis effort. However, documentation [29] and the disk [31] are available for future use. They can provide the greatest control of the TARGA 8 board.

The utilities can be found on two diskettes, "Truevision Trutilities" [32] and the "TARGA Software Tools Utilities Disk" [33]. The Trutilities disk includes, amongst others, programs to convert TARGA files to Encapsulted PostScript (.EPS) file format and Tagged Information File Format (.TIF). The .EPS file format can be imported to LATEX and the .TIF file format can be imported to Word Perfect 5.0. Unfortunately, these two programs have problems.

After running TRUEPS.EXE to convert to .EPS format, the last two command lines must be removed to make the file useful in LATEX. This can be done using any ASCII editor [35].

The program TRUETIF.EXE creates files that are not properly read by Word Perfect 5.0. Which program is incorrect is not clear. Details of this problem, as well as its solution can be found in Appendix F.

The utilities on the "TARGA Software Tools Utilities Disk" include the programs TARGAGET.EXE and TESTTARG.EXE. TARGAGET.EXE simply displays a TARGA file on the video monitor. While the very useful TESTTARG.EXE allows for user interactive control of the TARGA board.

To allow for easy use of the program TARGAGET.EXE a batch file was written, and kept in the root directory of the PC. The user could simply type "display filename.ext" at the C> prompt to display a file on the video monitor. Where filename.ext is the complete path of the TARGA file of interest. The batch file is listed below:

```
:DISPLAY.BAT
:This batch allows display of a TARGA file from any
:directory by executing TARGAGET after initializing the
:TARGA variables notice that a filename must be
:specified when TARGAGET is invoked (the %1 is the
:clue) likewise for this batch the proper DOS command
:would be C>display path\filename.ext

echo off
cls
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
targaget %1
```

Notice that since, in general, the TARGA files were stored in the \TARGA directory, only the filename and extension needed to be typed using this batch saving the keystrokes of the directory name. Notice also the setting of the two environmental variables, TARGA and TARGASET.

TESTTARG.EXE was by far the most useful program made available for this thesis. Although it was originally written to be used interactively through the keyboard, DOS allows for redirection of program execution. In other words, a file containing all desired keyboard entries could be specified when executing a program. By convention, these files have the extension .CMD.

At the C> prompt, typing "TESTTARG <path\filename.cmd" would execute the program TESTTARG using the command file specified by path\filename.cmd. The instructions listed in the .CMD file for the control of the TARGA board would then be executed as if the user were typing them in after TESTTARG prompts. For better understanding, running TESTTARG interactively is suggested. Help is available.

Another useful batch file LIVE.BAT puts the TARGA board in live mode by executing TESTTARG using the command file <live.cmd. BOTH files are provided below:

```
:LIVE.BAT
:this program simply puts the TARGA board in live
:camera mode.

echo off
cls
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <live.cmd
```

```
:LIVE.CMD
:This file will put the targa board in live mode when
:used as a batch with testtarg.exe.  The DOS command is
:"testtarg <live.cmd".

live
graphend
quit
```

A number of other batch and .CMD files were written for this thesis. They will be presented in Appendix G.

# APPENDIX C. FLIR DATA

This appendix presents information relating to the FLIR data used in the experiments and simulations of this thesis. First a brief description of the file formats will be presented, followed by a discussion of software used to alter these files. Computer code written during this thesis effort in "C" for Borland's Turbo-C compiler is also included.

## C.1 FLIR File Formats

The FLIR files used during this thesis have been at AFIT for some time, and have been used for a number of pattern recognition thesis efforts. In general these files can be found on the VMS machines throughout the Engineering school on various accounts, most recently Kevin Ayer's [24]. Originally these files were 240 pixel rows by 640 pixel columns, with intensities encoded with eight bits of grey scale in unsigned byte format. The files had no header information included, and the data was listed row major starting from the top left (unlike TARGA which started from bottom left).

Unsigned byte files in the VMS system require certain control characters to be present along with the data. For that reason most of the files had 644 bytes per row instead of the expected 640. The four extra bytes were at the end of each row.

These files may also have a different set of control characters which will appear as two extra bytes preceding each row, resulting in 642 bytes per row (possibly due to file transfer in other than binary mode). A third file type would have no control bytes, because they were removed using available software on the VMS system. The net result was that at least three distinct file types for the FLIR data might be encountered.

The solution to the problem was to create three separate programs for conversion of these files to TARGA format. After file transfer (using FTP) of these files to a PC, it was necessary to determine which file type was present. This could be determined by checking the file size. File sizes for zero, two, and four control bytes per row were 153600, 154080, and 154560 respectively.

### C.2 FLIR Conversion Software

The first step for preparing the FLIR files for display on the MOSLM was to convert them to TARGA format. Close inspection of the files showed that even numbered rows were significantly darker than there odd neighbors, probably due to something relating to interleaving in the original data collection process. This problem could be eliminated by averaging a 2X2 pixel area into 1 pixel.

A second problem was that the MOSLM had two defective rows. To avoid these, the files could have no more than 71 pixel rows, so targetless regions at the top and bottom of the images would not be converted. The final result of this first conversion step then was to convert the FLIR files to a TARGA file that was 71X320 pixels.

The three programs that accomplished this were FLR2HLF.C, FLR2HLF2.C, and FLR2HLF3.C for the zero, two and four control byte file types respectively. These programs were written so that it would be easy to write a .CMD file to convert a number of files successively, leaving the operator to pursue other things. All source code appears at the end of this appendix.

After converting the files to TARGA format, the next step was to binarize them. Ultimately the files would need to be 45X128 pixel, binary images for use in the JTC using the MOSLM. The technique used to perform this binarization was to binarize on some threshold that was a linear multiple of the mean value of the entire scene and "AND" this image with one that was binarized on a local 3X3 average. The user could select which 45 rows of the 71 available he would like to keep. Four 128 column files overlapping by 64 columns each would account for the entire 320 pixel columns of the reduced FLIR files. The programs used to accomplish this step were called HLF2AND.C and HLF2AND2.C, and like the FLR2HLF programs, they were made to be run with a .CMD file. The differences between HLF2AND.C and HLF2AND2.C were that the four files created by HLF2AND.C used the scene average threshold and had extensions 1, 2, 3 and 4, while HLF2AND2.C used a user specified scene threshold and had extensions 1a, 2a, 3a, and 4a.

The last program in this appendix was one that helped to create a template file from one of the 45X128 scene files. The program TEMPLATE.C allowed the user to select a 29X65 pixel segment within a scene file where he thought the target was located. By running the batch file TEMPLATE.BAT, which used the TESTTARG command file "template.cmd" to display the scene and template file, the results of this segmentation could be seen. Further background reduction could be accomplished using a hex editor (like the one available with Norton's Utilities), to hand segment the template.

The file "template.cmd" is created each time TEMPLATE.C is run and is different each time. However, TEMPLATE.BAT is presented below, followed by the source code for the above mentioned programs.

```
:TEMPLATE.BAT
:This batch allows the creation of a template file
:from a binary scene file for display on a MOSLM for
:use in the JTC.  The program TEMPLATE.C (a program
:written in Turbo-C) creates the file, and using the
:the command file2 "template.cmd" the program TESTTARG
:is used to display the original scene file and the
:resultant template on the video monitor.

:preparation
echo off
cls
SET TARGA=247
SET TARGASET=T3X-10G

CD\TURC
template
CD\TARGA
testtarg <template.cmd
```

```
/*****************************************************************/
/* FLR2HLF.C                                                     */
/*                                                               */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY              */
/*          CONVERSION PROGRAM FOR FLIR TO TARGA FILE FORMAT     */
/*                      by Capt John Cline                       */
/*                        July 31, 1989                          */
/*                                                               */
/*This program CONVERTS a user specified ~153k FLIR image file to */
/*~22k TARGA format by deleting the first and last 49 rows and   */
/*combining the remaining rows and columns 2 for 1.  The resulting */
/*file is 71x220 pixels, whereas the original was 240x640 pixels. */
/*To make these files useful for the SIM, it is necessary to run the */
/*program HLF2AND.C or HLF2AND2.C after running this one.  This   */
/*of the FLIR conversion program assumes NO CONTROL CHARACTERS will */
/*be present in the FLIR file.  If the FLIR files contain control */
/*characters, FLR2HLF2.C or FLR2HLF3.C must be run instead.      */
/*****************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, n, number_files;
char filename[256], filename2[256], filename3[256];
unsigned char bufferin[72][640];
unsigned char bufferout[36][320];
unsigned holder;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x40', '\x01', '\x47', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
 printf("Enter the number of files to process:  ");
 scanf("%u", &number_files);
 for(n=0; n<number_files; n++)
 {
   printf("FILE NUMBER %d OF %d\n", n+1, number_files);

   /*get input file name*/
   printf("Enter file name (no .flr) of FLIR file");   /*.flr assumed*/
   printf("in \TARGA directory).\n");        /*TARGA directory assumed*/
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s.flr",filename);
   sprintf(filename3,"\\targa\\%s.hlf",filename);

   /*open files and write header*/
   file2=fopen(filename3,"wb");
   fwrite(targhead,18,1,file2);
   file1=fopen(filename2,"rb");
```

```
    /*read data, invert rows, average 2 for 1, and write data*/
    fseek(file1, 77440, 0);
    fread(bufferin, 1, 44800, file1);
    for(j=0; j<70; j+=2)
    {
       for(k=0; k<640; k+=2)
       {
          holder=(bufferin[j][k]+bufferin[j+1][k]
             +bufferin[j][k+1]+bufferin[j+1][k+1]+2)/4;
          bufferout[34-j/2][k/2]=holder;
       }
    }
    fwrite(bufferout, 11200, 1, file2);

    fseek(file1, 31360, 0);
    fread(bufferin, 1, 46080, file1);
    for(j=0; j<72; j+=2)
    {
       for(k=0; k<640; k+=2)
       {
          holder=(bufferin[j][k]+bufferin[j+1][k]
             +bufferin[j][k+1]+bufferin[j+1][k+1]+2)/4;
          bufferout[35-j/2][k/2]=holder;
       }
    }
    fwrite(bufferout, 11520, 1, file2);
    fclose(file1);
    fclose(file2);
  }
 }
/**********************************************************************/
```

```
/*****************************************************************************/
/* FLR2HLF2.C                                                               */
/*                                                                          */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                       */
/*           CONVERSION PROGRAM FOR FLIR TO TARGA FILE FORMAT               */
/*                        by Capt John Cline                                */
/*                           July 31, 1989                                  */
/*                                                                          */
/*This program CONVERTS a user specified ~153k FLIR image file to           */
/*~22k TARGA format by deleting the first and last 49 rows and              */
/*combining the remaining rows and columns 2 for 1.  The resulting          */
/*file is 71x320 pixels, whereas the original was 240x640 pixels.           */
/*To make these files useful for the SLM, it is necessary to run the        */
/*program HLF2AND.C or HLF2AND2.C after running this one.  This             */
/*version of the FLIR conversion program assumes 2 LEADING BYTES of         */
/*control characters will be present in the FLIR file.  If the FLIR         */
/*files contain 0 or 4 control characters, FLR2HLF.C or FLR2HLF3.C          */
/*must be run instead.                                                      */
/*****************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, n, number_files;
char filename[256], filename2[256], filename3[256];
unsigned char bufferin[72][642];
unsigned char bufferout[36][320];
unsigned holder;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x40', '\x01', '\x47', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
 printf("Enter the number of files to process:  ");
 scanf("%u", &number_files);
 for(n=0; n<number_files; n++)
  {
   printf("FILE NUMBER %d OF %d\n", n+1, number_files);

   /*get input file name*/
   printf("Enter file name (no .flr) of FLIR file");   /*.flr assumed*/
   printf("in \TARGA directory).\n");           /*TARGA directory assumed*/
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s.flr",filename);
   sprintf(filename3,"\\targa\\%s.hlf",filename);

   /*open files and write header*/
   file2-fopen(filename3,"wb");
   fwrite(targhead,18,1,file2);
   file1=fopen(filename2,"rb");

   /*read data, invert rows, average 2 for 1, and write data*/
   fseek(file1, 77682, 0);
   fread(bufferin, 1, 44940, file1);
   for(j=0; j<70; j+=2)
   {
      for(k=2; k<642; k+=2)
      {
         holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
            +bufferin[j+1][k+1]+2)/4;
         bufferout[34-j/2][k/2-1]=holder;
      }
   }
   fwrite(bufferout, 11200, 1, file2);

   fseek(file1, 31458, 0);
   fread(bufferin, 1, 46224, file1);
   for(j=0; j<72; j+=2)
   {
      for(k=2; k<642; k+=2)
      {
         holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
            +bufferin[j+1][k+1]+2)/4;
         bufferout[35-j/2][k/2-1]=holder;
      }
   }
   fwrite(bufferout, 11520, 1, file2);
   fclose(file1);
   fclose(file2);
 }
}
/**********************************************************************/
```

```
/*********************************************************************/
/* FLR2HLF3.C                                                        */
/*                                                                   */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/*         CONVERSION PROGRAM FOR FLIR TO TARGA FILE FORMAT          */
/*                      by Capt John Cline                           */
/*                         July 31, 1989                             */
/*                                                                   */
/*This program CONVERTS a user specified ~153k FLIR image file to    */
/*~22k TARGA format by deleting the first and last 49 rows and       */
/*combining the remaining rows and columns 2 for 1.  The resulting   */
/*file is 71x320 pixels, whereas the original was 240x640 pixels.    */
/*To make these files useful for the SLM, it is necessary to run the */
/*program HLF2AND.C or HLF2AND2.C after running this one.  This      */
/*of the FLIR conversion program assumes 4 CONTROL CHARACTERS will   */
/*be present at the end of each row in the FLIR file.  If the FLIR   */
/*files contain zero or two control characters, FLR2HLF.C or         */
/*FLR2HLF2.C must be run instead.                                    */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, n, number_files;
char filename[256], filename2[256], filename3[256];
unsigned char bufferin[72][644];
unsigned char bufferout[36][320];
unsigned holder;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x40', '\x01', '\x47', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
 printf("Enter the number of files to process:  ");
 scanf("%u", &number_files);
 for(n=0; n<number_files; n++)
  {
   printf("FILE NUMBER %d OF %d\n", n+1, number_files);

   /*get input file name*/
   printf("Enter file name (no .flr) of FLIR file");    /*.flr assumed*/
   printf("in \TARGA directory).\n");         /*TARGA directory assumed*/
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s.flr",filename);
   sprintf(filename3,"\\targa\\%s.hlf",filename);

   /*open files and write header*/
   file2=fopen(filename3,"wb");
   fwrite(targhead,18,1,file2);
   file1=fopen(filename2,"rb");

   /*read data, invert rows, average 2 for 1, and write data*/
   fseek(file1, 77924, 0);
   fread(bufferin, 1, 45080, file1);
   for(j=0; j<70; j+=2)
   {
      for(k=0; k<640; k+=2)
      {
         holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
            +bufferin[j+1][k+1]+2)/4;
         bufferout[34-j/2][k/2]=holder;
      }
   }
   fwrite(bufferout, 11200, 1, file2);

   fseek(file1, 31556, 0);
   fread(bufferin, 1, 46368, file1);
   for(j=0; j<72; j+=2)
   {
      for(k=0; k<640; k+=2)
      {
         holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
            +bufferin[j+1][k+1]+2)/4;
         bufferout[35-j/2][k/2]=holder;
      }
   }
   fwrite(bufferout, 11520, 1, file2);
   fclose(file1);
   fclose(file2);
 }
}
/***********************************************************************/
```

```
/*********************************************************************/
/* HLF2AND.C                                                         */
/*                                                                   */
/*                 AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*     PROGRAM TO BINARIZE FLIR .HLF FILES FOR USE WITH THE SLM      */
/*                       by Capt John Cline                          */
/*                          July 31, 1989                            */
/*                                                                   */
/*This program creates a BINARY version of TARGA files, that were    */
/*originally FLIR data.  It should be run after running FLR2HLF.C,   */
/*FLR2HLF2.C, or FLR2HLF3.C which reduce the original 240x640 FLIR   */
/*file to 71x320 pixels.  The program first binarizes the pixel      */
/*values based on the 71X320 image's AVERAGE intensity.  It also     */
/*binarizes based on using the local average of 3x3 pixels.  Both    */
/*conditions must be satisfied for the pixel to receive a value of   */
/*'1'.  Finally, the program breaks the file up into four smaller    */
/*45X128 files for use with the SLM in the JTC.  The user specifes   */
/*offset that determines which of the 71 rows will be included.  The */
/*files are stored in TARGA format in the files "filename.1" thru    */
/*"filename.4".  To use these files on the SLM, you must run the     */
/*program DIS-HF.C.  Another program, HLF2AND2.C allows you to       */
/*specify a multiplier for the scene average threshold.              */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

float threshold=0, thrshld;
unsigned int jj, i, j, k, l, m, offset, n, number_files;
char filename[256], filename2[256];
unsigned char bufferin[45][320];
unsigned char bufferout1[45][320], bufferout2[45][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x2d', '\x00', '\x08', '\x00'};

FILE *file1;
void main()
{
 printf("Enter a vertical offset (integer between 0 and 26):\n");
 scanf("%u", &offset);
 offset=320*offset+18;
 printf("Enter the number of files to process:  ");
 scanf("%u", &number_files);
 for(n=0; n<number_files; n++)
 {
   printf("FILE NUMBER %d OF %d\n", n+1, number_files);

   /*get file name */
   printf("Enter file name (no .hlf)");                  /*assume .hlf*/
   printf(" (in \TARGA directory.)\n");                 /*assume \TARGA*/
   scanf("%s", filename);
```

```
sprintf(filename2,"\\targa\\%s.hlf",filename);

/*read in data*/
printf("\nreading in data\n");
file1=fopen(filename2,"rb");
fseek(file1, offset, 0);
fread(bufferin, 1, 14400, file1);
fclose(file1);

/*Do the scene average threshold*/
printf("calculating scene average intensity\n");
for(i=0; i<45; i++)
{
    for(j=0; j<320; j++)
    {
        threshold+=bufferin[i][j];
    }
}
threshold=threshold/14400;

/*Now do the local 3X3*/
printf("calculating local 3x3 average intensity\n count to 43\n");
for(i=1; i<44; i++)
{
    printf("%u\n",i);
    for(j=1; j<319; j++)
    {
        thrshld=0;
        if(bufferin[i][j]<=threshold) bufferout1[i][j]=0x00;
        else
        {
            for(l=i-1; l<i+2; l++)
            {
                for(m=j-1; m<j+2; m++)
                {
                    thrshld+=bufferin[l][m];
                }
            }
            if(bufferin[i][j]<=thrshld/9) bufferout1[i][j]=0x00;
            else bufferout1[i][j]=0xff;
        }
    }
}

/*zero edges of output array*/
for(i=0; i<45; i++)
{
    bufferout1[i][0]=0x00;
    bufferout1[i][319]=0x00;
}
for(j=0; j<320; j++)
{
    bufferout1[0][j]=0x00;
```

```
      bufferout1[44][j]=0x00;
   }

   /*write the four files*/
   printf("writing output files");
   for(k=0; k<4; k++)
   {
      for(i=0; i<45; i++)
      {
         jj=k*64;
         for(j=0; j<128; j++)
         {
            bufferout2[i][j]=bufferout1[i][jj];
            jj++;
         }
      }
      sprintf(filename2,"\\targa\\%s.%u",filename,k+1);
      file1=fopen(filename2,"wb");
      fwrite(targhead,18,1,file1);
      fwrite(bufferout2,5760,1,file1);
      fclose(file1);
   }
 }
}
/************************************************************************/
```

```
/***********************************************************************/
/* HLF2AND2.C                                                          */
/*                                                                     */
/*                 AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*     PROGRAM TO BINARIZE FLIR .HLF FILES FOR USE WITH THE SLM        */
/*                       by Capt John Cline                            */
/*                          July 31, 1989                              */
/*                                                                     */
/*This program creates a BINARY version of TARGA files, that were      */
/*originally FLIR data.  It should be run after running FLR2HLF.C,     */
/*FLR2HLF2.C, or FLR2HLF3.C which reduce the original 240x640 FLIR     */
/*file to 71x320 pixels.  The program first binarizes the pixel        */
/*values based on the 71X320 image's AVERAGE times a user specified    */
/*multiplier.  It also binarizes based on using the local average of   */
/*3x3 pixels.  Both conditions must be satisfied for the pixel to      */
/*receive a value of '1'.  Finally, the program breaks the file up     */
/*into four smaller 45X128 files for use with the SLM in the JTC.      */
/*The user specifes an offset that determines which of the 71 rows     */
/*will be included.  The files are stored in TARGA format in the       */
/*files "filename.1a" thru "filename.4a".  To use these files on the   */
/*SLM, you must run the program DIS-HF.C.  Another program,            */
/*HLF2AND.C, does not allow a multiplier for the scene average         */
/*threshold and stores the files in "filename.1" thru "filename.4".    */
/***********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

float multiplier, threshold=0, thrshld;
unsigned int jj, i, j, k, l, m, offset, n, number_files;
char filename[256], filename2[256];
unsigned char bufferin[45][320];
unsigned char bufferout1[45][320], bufferout2[45][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x2d', '\x00', '\x08', '\x00'};

FILE *file1;
void main()
{
 printf("Enter a vertical offset (integer between 0 and 26):\n");
 scanf("%u", &offset);
 offset=320*offset+18;
 printf("ENTER threshold multiplier (mean=1;");
 printf(" 1.5 times mean=1.5 etc.):\n");
 scanf("%f", &multiplier);
 printf("Enter the number of files to process:  ");
 scanf("%u", &number_files);
 for(n=0; n<number_files; n++)
 {
   printf("FILE NUMBER %d OF %d\n", n+1, number_files);
```

```
/*get file name */
printf("Enter file name (no .hlf)");                /*assume .hlf*/
printf(" (in \TARGA directory.)\n");              /*assume \TARGA*/
scanf("%s", filename);
sprintf(filename2,"\\targa\\%s.hlf",filename);

/*read in data*/
printf("\nreading in data\n");
filel=fopen(filename2,"rb");
fseek(filel, offset, 0);
fread(bufferin, 1, 14400, filel);
fclose(filel);

/*Do the scene average threshold*/
printf("calculating scene average intensity\n");
for(i=0; i<45; i++)
{
   for(j=0; j<320; j++)
   {
      threshold+=bufferin[i][j];
   }
}
threshold=multiplier*threshold/14400;

/*Now do the local 3X3*/
printf("calculating local 3x3 average intensity\n count to 43\n");
for(i=1; i<44; i++)
{
   printf("%u\n",i);
   for(j=1; j<319; j++)
   {
      thrshld=0;
      if(bufferin[i][j]<=threshold) bufferoutl[i][j]=0x00;
      else
      {
         for(l=i-1; l<i+2; l++)
         {
            for(m=j-1; m<j+2; m++)
            {
               thrshld+=bufferin[l][m];
            }
         }
         if(bufferin[i][j]<=thrshld/9) bufferoutl[i][j]=0x00;
         else bufferoutl[i][j]=0xff;
      }
   }
}

/*zero edges of output array*/
for(i=0; i<45; i++)
{
   bufferoutl[i][0]=0x00;
   bufferoutl[i][319]-0x00;
   }
   for(j=0; j<320; j++)
   {
      bufferout1[0][j]=0x00;
      bufferout1[44][j]=0x00;
   }

   /*write the four files*/
   printf("writing output files");
   for(k=0; k<4; k++)
   {
      for(i=0; i<45; i++)
      {
         jj=k*64;
         for(j=0; j<128; j++)
         {
            bufferout2[i][j]=bufferout1[i][jj];
            jj++;
         }
      }
      sprintf(filename2,"\\targa\\%s.%ua",filename,k+1);
      file1=fopen(filename2,"wb");
      fwrite(targhead,18,1,file1);
      fwrite(bufferout2,5760,1,file1);
      fclose(file1);
   }
 }
}
/*********************************************************************/
```

```
/*********************************************************************/
/* TEMPLATE.C                                                        */
/*                                                                   */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                */
/*          PROGRAM FOR CREATING TEMPLATES FOR USE IN THE JTC        */
/*                        by Capt John Cline                         */
/*                           July 31, 1989                           */
/*                                                                   */
/*This program will create a ~7k 45x128 pixel TARGA file using 29x65 */
/*pixels from a user specified binary FLIR file in TARGA format      */
/*created by the program HLF2AND.C or HLF2AND2.C.  The user selects  */
/*the coordinates of the center pixel in the original file, and the  */
/*29x65 pixels will be written onto a black background centered at   */
/*pixel location (64, 14).  The template file can be displayed on    */
/*SLM using the program DIS-HF which is part of a joint transform    */
/*correlator.  The program also creates a .CMD file for use with     */
/*the TARGA program TESTTARG which allows the template and scene     */
/*file to be displayed on the video monitor after template creation. */
/*The batch file TEMPLATE.BAT will perform all necessarry steps      */
/*except some further segmentation of the template that must be done */
/*use a hex editor like the one available with Norton's Utilities.   */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

unsigned int i, j, ii, jj, xc, yc;
char filename[256], filename2[256], filename3[256];
unsigned char bufferin[45][128];
unsigned char bufferout[45][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x2d', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   printf("Enter file name and .ext of AND file");
   printf(" (in \TARGA directory).\n");                /*assume \TARGA*/
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s",filename);
   printf("reading in data\n");
   file1=fopen(filename2,"rb");
   fseek(file1,18,0);
   fread(bufferin, 1, 5760, file1);
   fclose(file1);

   /*template will go in TARGA directory with .tmp extension*/
   printf("Enter file name (no .tmp)");
   printf(" of template file (to \TARGA directory).\n");
   scanf("%s", filename);
```

```
    sprintf(filename3,"\\targa\\%s.tmp",filename);
    file2=fopen(filename3,"wb");

    /*create .cmd file*/
    file1=fopen("\\targa\\template.cmd","w");
    fprintf(file1,"live\nerase\n0\n0\n0\ndis\ngetpic\n%s",filename2);
    fprintf(file1,"\n0\n0\n-1\ngetpic\n%s.tmp\n0\n50\n-1\n",filename);
    fprintf(file1,"graphend\nquit\n");
    fclose(file1);

    /*create template file*/
    printf("Enter center coordinates xc between 32 and 95 inclusive\n");
    printf("and yc between 14 and 30 inclusive:\n");
    printf("xc:");
    scanf("%u", &xc);
    printf("yc:");
    scanf("%u", &yc);
    ii=0;
    for(i=yc-14; i<yc+15; i++)
    {
       jj=32;
       for(j=xc-32; j<xc+33; j++)
       {
          bufferout[ii][jj]=bufferin[i][j];
          jj++;
       }
       ii++;
    }
    fwrite(targhead,18,1,file2);
    fwrite(bufferout, 5760, 1, file2);
    fclose(file2);
}
/**************************************************************/
```

# APPENDIX D. SPIRICON FRAMEGRABBER

This appendix presents information relating to the Spiricon framegrabber system. It is included for two reasons. First, this system can provide 3D plotting of its files that enhance documentation. Second, this camera and framegrabber system may be used in future related research. The manual can provide more information concerning this system if it is needed [36].

A brief discussion of Word Perfect 5.0 import of these 3D plots and description of the file formats will be presented, followed by a discussion of software used to convert TARGA file formats to Spiricon and vice versa. Computer code written during this thesis effort in "C" for Borland's Turbo-C compiler is also included.

## D.1 Documentation in Word Perfect 5.0

The following discussion assumes knowledge of Word Perfect 5.0's graphic program GRAB.COM. 3D plots using the Spiricon system can be imported into Word Perfect 5.0 if a few steps are followed. GRAB.COM must be run first, before executing the Spiricon software. GRAB.COM, which can convert the computer monitor display into a graphic file "grabX.wpg" for later import into Word Perfect 5.0, is normally easy to use. However, when it runs along with the Spiricon software some strange things happen.

The first problem encountered is that the outline box is not visible, but the correct region is usually captured if no cursor buttons are pressed. The second problem is that after the file is created, the computer locks up and must be reset by powering down and back up again (Ctrl-Alt-Del won't work). This may be too tedious if a number of plots are necessary, so cut and paste may be preferred using Spiricon's built in print function.

## D.2 Spiricon File Formats

Spiricon files are 512X512 arrays of eight bit grey scale pixels in unsigned byte format. The only differences, other than the size of the file (512X512 versus 400X512), between Spiricon and TARGA files are the necessary header information, and that the TARGA files are row major starting from the lower left, while the Spiricon files are row major starting from the upper left.

Instead of the 18 byte header used by TARGA files, Spiricon files have only two bytes preceding the image data, hex 39 and hex 30. Spiricon files also have 906 bytes of information at the end of each file relating to equipment configuration. So the total length of the Spiricon file is 263052 bytes [36:79], versus 204818 bytes for a full screen (400X512) TARGA file.

## D.3 Spiricon Conversion Software

The first of three conversion programs, TAR2SPIR.C, converts a 400X512 TARGA file to a 512X512 Spiricon file by adding 56 rows of zeros both above and below the image data.

The 906 bytes of information typically at the end of each Spiricon file is read from the file "spir.fig" that is used by the Spiricon system software, and can be found on the distribution diskettes [37].

The second program, SPIR2TAR.C, converts the Spiricon file to a 400X512 file by deleting the first and last 56 rows. A third program, SPI2TAR.C, creates a 512X512 TARGA file which can not be fully viewed on the TARGA video monitor. However, this does allow Spiricon files to be converted to Encapsulated PostScript files for use with Latex (see Appendix B).

```
/*******************************************************************/
/* TAR2SPIR.C                                                      */
/*                                                                 */
/*             AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*      CONVERSION PROGRAM FOR TARGA TO SPIRICON FILE FORMAT       */
/*                   by Capt John Cline                            */
/*                     July 31, 1989                               */
/*                                                                 */
/*This program CONVERTS a user specified TARGA file to Spiricon file */
/*format by adding 56 blank rows above and below the image data.   */
/*The file "spir.fig" must be present to provide header information */
/*for the Spiricon file.  Since Spiricon's files start from the top */
/*TARGA's start from the bottom, inversion is necessary.           */
/*******************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k;
char filename[256], filename2[256], holder[512];
char bufferit[100][512];
char spirhead[2]={'\x39','\x30'};
char spirtail[906];

FILE *file1, *file2, *file3;

void main()
{
   /*get file names and open files*/
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of SPIRICON file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*write 2 header bytes and 56 blank rows*/
   fwrite(spirhead, sizeof(spirhead), 1, file2);
   fwrite(bufferit, 28672, 1, file2);

   /*read TARGA, invert rows and write Spiricon*/
   printf("Count to 4\n");
   for(i=0; i<4; i++)
   {
      printf("%d\n",i+1);
      fseek(file1, 18+51200*(3-i), 0);
      fread(bufferit, 1, 51200, file1);
      for(j=0; j<50; j++)
      {
         for(k=0; k<512; k++)
         {
            holder[k]=bufferit[j][k];
```

```
                bufferit[j][k]=bufferit[99-j][k];
                bufferit[99-j][k]=holder[k];
            }
        }
        fwrite(bufferit, sizeof(bufferit), 1, file2);
    }
    fclose(file1);

    /*write 56 blank rows*/
    for(i=0; i<512; i++)
    {
        for(j=0; j<56; j++)
        {
            bufferit[j][i]=0x00;
        }
    }
    fwrite(bufferit, 28672, 1, file2);

    /*write configuration file to Spiricon file*/
    file3=fopen("spir.fig","rb");
    fread(spirtail, 1, 906, file3);
    fclose(file3);
    fwrite(spirtail, sizeof(spirtail), 1, file2);
    fclose(file2);
}
/*********************************************************************/
```

```
/*********************************************************************/
/* SPIR2TAR.C                                                        */
/*                                                                   */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/*       CONVERSION PROGRAM FOR SPIRICON TO TARGA FILE FORMAT        */
/*                      by Capt John Cline                           */
/*                         July 31, 1989                             */
/*                                                                   */
/*This program CONVERTS a user specified Spiricon file to TARGA file */
/*format by deleting 56 rows on the top and bottom of the image data.*/
/*An 18 byte header is added to the TARGA file.  Since Spiricon's    */
/*files start from the top and TARGA's start from the bottom,        */
/*inversion is necessary.  The program SPI2TAR.C will create a       */
/*512X512 TARGA file (no deleted lines), however it can not be viewed*/
/*in its entirity on the video monitor.                              */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k;
char filename[256], filename2[256], holder[512];
char bufferit[100][512];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x00', '\x02', '\x90', '\x01', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   /*get filenames and open files*/
   printf("Enter complete path and file name of SPIRICON file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*write header*/
   fwrite(targhead, sizeof(targhead), 1, file2);

   /*read Spiricon, invert files and write TARGA*/
   printf("Count to 4\n");
   for(i=0; i<4; i++)
   {
      printf("%d\n",i+1);
      fseek(file1, 28674+(3-i)*51200, 0);
      fread(bufferit, 1, 51200, file1);

      for(j=0; j<50; j++)
      {
         for(k=0; k<512; k++)
         {
            holder[k]=bufferit[j][k];
            bufferit[j][k]=bufferit[99-j][k];
            bufferit[99-j][k]=holder[k];
         }
      }
   fwrite(bufferit, sizeof(bufferit), 1, file2);
   }
   fclose(file1);
   fclose(file2);
}
```

```
/**********************************************************************/
/* SPI2TAR.C                                                          */
/*                                                                    */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*         CONVERSION PROGRAM FOR SPIRICON TO TARGA FILE FORMAT       */
/*                       by Capt John Cline                           */
/*                          July 31, 1989                             */
/*                                                                    */
/*This program CONVERTS a user specified Spiricon file to TARGA file */
/*format without deleting any rows.  The resultant TARGA file is     */
/*512X512 pixels 112 rows too large for display of the entire image  */
/*on the TARGA video monitor.  However, this step allows the Spiricon*/
/*files to be converted to Encapsulated PostScript using the TARGA   */
/*program TRUEPS.EXE.  The resultant PostScript file must have its   */
/*last two command lines removed using an ASCII editor to be imported*/
/*into Latex.  An 18 byte header is added to the TARGA file; and     */
/*since Spiricon's files start from the top and TARGA's start from   */
/*the bottom, inversion is necessary.  The program SPIR2TAR.C will   */
/*create a 400X512 TARGA file (with deleted lines) that can be viewed*/
/*in its entirity on the video monitor.                              */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k;
char filename[256], filename2[256], holder[512];
char bufferit[64][512];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x00', '\x02', '\x00', '\x02', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   /*get filenames and open files*/
   printf("Enter complete path and file name of SPIRICON file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*write header*/
   fwrite(targhead, sizeof(targhead), 1, file2);

   /*read Spiricon, invert files and write TARGA*/
   printf("Count to 8\n");
   for(i=0; i<8; i++)
   {
      printf("%d\n",i+1);
      fseek(file1, 2+(7-i)*32768, 0);
      fread(bufferit, 1, 32768, file1);

      for(j=0; j<32; j++)
      {
         for(k=0; k<512; k++)
         {
            holder[k]=bufferit[j][k];
            bufferit[j][k]=bufferit[63-j][k];
            bufferit[63-j][k]=holder[k];
         }
      }
      fwrite(bufferit, sizeof(bufferit), 1, file2);
   }
   fclose(file1);
   fclose(file2);
}
```

# APPENDIX E. IMAGING TECHNOLOGY SYSTEM

This appendix presents information relating to the Imaging Technology image processing system. This system was made available by ASD\ENAML, and was used to perform the simulations of Chapter III. The equipment can perform a number of functions including 2D Fourier transforming and image rotation, which may be useful for future attempts at performing rotation invariant pattern recognition. Additional information can be acquired from the system manuals [38] or through personnel at ASD\ENAML [39].

A description of the file formats for the Imaging Technology system is presented, followed by a discussion of the software used for file conversion. Computer code written during this thesis effort in "C" for Borland's Turbo-C compiler is also included.

## E.1 Imaging Technology File Format

All Imaging Technology files have a .img extension in their DOS file name. The largest Imaging Technology files are 480X512 arrays of eight bit grey scale pixels in unsigned byte format. The only differences, other than the maximum size of the file (480X512 versus 400X512), between Imaging Technology and TARGA files are the necessary header information, and that the TARGA files are row major starting from the lower left, while the Imaging Technology files are row major starting from the upper left.

Instead of the 18 byte header used by TARGA files, Imaging Technology files can have a variable header size, depending upon the length of the comment field inside the header. The header specifications for Imaging Technology files are listed in the system manuals [38:C-1] and are reproduced below.

| Bytes | Contents |
|---|---|
| 0-1 | Characters IM indicate this is an image file |
| 2-3 | Comment length |
| 4-5 | Width of the image in pixels |
| 6-7 | Height of the image in lines |
| 8-9 | Coordinates of original X-axis position |
| 10-11 | Coordinates of original Y-axis position |
| 12-13 | File type flag 0 = 8 bit<br>1 = compressed<br>4 = 16 bit |
| 14-63 | Reserved |
| 64-n | Comment area - variable in length; maximum 255 bytes |
| rest | Data area |

## E.2 Imaging Technology Conversion Software

The first of three conversion programs, TAR2IMG.C, converts any size TARGA file to a Imaging Technology file by changing the header and inverting the data to read from top down instead of from bottom up.

The second program, IMG2TAR.C, converts Imaging Technology files of any size to TARGA file format by reversing the process of TAR2IMG.C. This may result in a TARGA file that is too large to be fully viewed on the TARGA video monitor. However, this does allow Imaging Technology files to be converted to Encapsulated PostScript files for use with Latex (see Appendix B).

The third program, IMG2TAR2.C, converts a 480X512 Imaging Technology file to a 400X512 TARGA file by deleting the last 80 lines.

```
/********************************************************************/
/* TAR2IMG.C                                                        */
/*                                                                  */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*  CONVERSION PROGRAM FOR TARGA TO IMAGING TECHNOLOGY FILE FORMAT  */
/*                      by Capt John Cline                          */
/*                      September 30, 1989                          */
/*                                                                  */
/*This program CONVERTS a user specified TARGA file to Imaging      */
/*Technology file format.  A 64 byte header is added to the Imaging */
/*Technology file; and since TARGA files start from the bottom and  */
/*Imaging Technology files start from the top, inversion is         */
/*necessary.                                                        */
/********************************************************************/

#include "stdio.h"
#include "string.h"

long int seek_length;
int i, columns, rows;
unsigned char filename[256], filename2[256], buffer[512], targhead[18];
char imtechhead[64]={'\x49','\x4d','\x00','\x00','\x00','\x02',
                     '\x90','\x01','\x00','\x00','\x00','\x00',
                     '\x00','\x00','\xd7','\x51','\x72','\xf9',
                     '\x1d','\x03','\xc4','\x1d','\x06','\x00',
                     '\x20','\x00','\x1b','\x00','\x89','\x55',
                     '\x5d','\x02','\xc4','\x1d','\x9a','\xf9',
                     '\x0a','\x01','\x82','\x00','\xfd','\xfa',
                     '\x9e','\xf9','\x33','\x00','\x5d','\x2c',
                     '\x96','\xf9','\x82','\x00','\xfd','\xfa',
                     '\x9e','\xf9','\x40','\x00','\x07','\x59',
                     '\x82','\x00','\x0f','\x00'};

FILE *file1, *file2;

void main()
{
   /*get filenames and open files*/
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name (and .img) of");
   printf(" IMAGING TECHNOLOGY file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*get TARGA input file specs and write Imaging Technology header*/
   fread(targhead, 1, 18, file1);
   columns=(int)targhead[12]+256*(int)targhead[13];
   rows=(int)targhead[14]+256*(int)targhead[15];
   imtechhead[4]=targhead[12];
   imtechhead[5]=targhead[13];
   imtechhead[6]=targhead[14];
   imtechhead[7]=targhead[15];
   fwrite(imtechhead,1,64,file2);

   /*read in TARGA and write Imaging Technology one row at a time*/
   for(i=rows-1; i>=0; i--)
   {
      seek_length = (long)i * (long)columns + 18;
      fseek(file1, seek_length, 0);
      fread(buffer, 1, columns, file1);
      fwrite(buffer, columns, 1, file2);
   }
   fclose(file1);
   fclose(file2);
   printf("done \n");
}
/*********************************************************************/
```

```
/**********************************************************************/
/* IMG2TAR.C                                                          */
/*                                                                    */
/*                    AIR FORCE INSTITUTE OF TECHNOLOGY               */
/*  CONVERSION PROGRAM FOR IMAGING TECHNOLOGY TO TARGA FILE FORMAT    */
/*                         by Capt John Cline                         */
/*                         September 30, 1989                         */
/*                                                                    */
/*This program CONVERTS a user specified Imaging Technology file to   */
/*TARGA file format without deleting any rows.  The resultant TARGA   */
/*file is a maximum of 480X512 pixels 80 rows too large for display   */
/*of the entire image on the TARGA video monitor.  However, this step*/
/*allows the Imaging Technology Files to be converted to Encapsulated*/
/*PostScript using the TARGA program TRUEPS.EXE.  The resultant       */
/*PostScript file must have its last two command lines removed using */
/*an ASCII editor to be imported into Latex.  An 18 byte header is    */
/*added to the TARGA file; and since Imaging Technology files start   */
/*from the top and TARGA's start from the bottom, inversion is        */
/*necessary.  The program IMG2TAR2.C will create a 400X512 TARGA file*/
/*from a 480X512 Imaging Technology file by deleting the last 80      */
/*lines, so that the resultant TARGA file can be viewed in its        */
/*entirity on the video monitor.                                      */
/**********************************************************************/

#include "stdio.h"
#include "string.h"

int i;
unsigned char holder[2];
char filename[256], filename2[256], buffer[512];
long int columns, rows, comment length, seek length;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x00', '\x02', '\x90', '\x01', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   /*get filenames and open*/
   printf("Enter complete path and file name of");
   printf(" IMAGING TECHNOLOGY file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*get IMAGING TECHNOLOGY input file specs and write TARGA header*/
   fseek(file1, 2, 0);
   fread(holder, 1, 2, file1);
   comment_length=holder[0]+256*holder[1];
   fread(holder, 1, 2, file1);
   columns=holder[0]+256*holder[1];
   targhead[12]=holder[0];
   targhead[13]=holder[1];
   fread(holder, 1, 2, file1);
   rows=holder[0]+256*holder[1];
   targhead[14]=holder[0];
   targhead[15]=holder[1];
   fwrite(targhead,1,18,file2);

   /*read in Imaging Technology and write TARGA one row at a time*/
   for(i=rows-1; i>=0; i--)
   {
      seek_length = (long)i * columns + 64 + comment_length;
      fseek(file1, seek_length, 0);
      fread(buffer, 1, columns, file1);
      fwrite(buffer, columns, 1, file2);
   }
   fclose(file1);
   fclose(file2);
   printf("done \n");
}
/********************************************************************/
```

```
/**************************************************************************/
/* IMG2TAR2.C                                                            */
/*                                                                       */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*  CONVERSION PROGRAM FOR IMAGING TECHNOLOGY TO TARGA FILE FORMAT       */
/*                         by Capt John Cline                            */
/*                         September 30, 1989                            */
/*                                                                       */
/*This program CONVERTS a user specified 480X512 Imaging Technology     */
/*file to a 400X512 TARGA file format by deleting the last 80 rows.     */
/*An 18 byte header is added to the TARGA file; and since Imaging       */
/*Technology files start from the top and TARGA's start from the        */
/*bottom, inversion is necessary.  The program IMG2TAR.C will create    */
/*TARGA files that match the size of the input Imaging Technology      */
/*file.                                                                  */
/**************************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, k;
unsigned wastelength;
char filename[256], filename2[256], wasteheader[200], holder[512];
unsigned commentlength;
char bufferit[100][512];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x00', '\x02', '\x90', '\x01', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   /*get file names and open files*/
   printf("Enter complete path and file name of");
   printf(" 480X512 IMAGE TECHNOLOGY file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and filename of 400X512 TARGA file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*write header*/
   fwrite(targhead, sizeof(targhead), 1, file2);

   /*get Imaging Technology file specs*/
   fread(wasteheader, 1, 4, file1);
   commentlength=wasteheader[2]+256*wasteheader[3];
   wastelength=64+commentlength;

   printf("Count to 4\n");
   for(i=0; i<4; i++)
   {
      printf("%d\n",i+1);
      fseek(file1, wastelength+51200*(3-i), 0);
      fread(bufferit, 1, 51200, file1);
      for(j=0; j<50; j++)
      {
         for(k=0; k<512; k++)
         {
            holder[k]=bufferit[j][k];
            bufferit[j][k]=bufferit[99-j][k];
            bufferit[99-j][k]=holder[k];
         }
      }
      fwrite(bufferit, sizeof(bufferit), 1, file2);
   }
   fclose(file1);
   fclose(file2);
}
/**********************************************************************/
```

# APPENDIX F. TAGGED IMAGE FILE FORMAT (TIFF)

This appendix presents information relating to the conversion of TARGA files to TIFF for import into Word Perfect 5.0. This technique created the grey scale imagery that is pictured in Chapters III, IV, and V. The software provided by the vendor, TRUETIF.EXE [32], did not interface well with Word Perfect 5.0. The resulting images in Word Perfect appeared to be shifted by several pixels. For that reason, software was written to correct the problem.

A discussion of the necessary fix actions is given below. Computer code written during this thesis effort in "C" for Borland's Turbo-C compiler is also included. Limited documentation is available for TIFF at AFIT [40].

## F.1 TIFF Conversion Software

By inspection of the various files created by TRUETIF.EXE, and through trial and error techniques, a little was learned about TIFF headers. The result was that a modified header file "tifhead.fil" was created. Using this header as a baseline for (400X512 images), only two observations were neccssary to make the header generic. The first was that the width in pixels was located in bytes 30 and 31, while the height in rows was located in bytes 42 and 43, so these four bytes had to be changed for different file sizes. The second was that the header indicated that the TIFF file would begin at the top left corner, so lack of

TIFF documentation dictated inverting the TARGA data.

Two programs appear below: TAR2TIF.C, which converts a 400X512 TARGA file to TIFF; and TAR2TIF2.C, which converts TARGA files of any size to TIFF. TAR2TIF.C runs faster!

One additional note: An error message appears in Word Perfect when these files are imported. Ignore it when it appears (by pressing [Enter]), and everything will work fine.

```
/**********************************************************************/
/* TAR2TIF.C                                                          */
/*                                                                    */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                */
/*  CONVERSION PROGRAM FOR TARGA TO TAGGED IMAGE FILE FORMAT (TIFF)   */
/*                         by Capt John Cline                         */
/*                          October 31, 1989                          */
/*                                                                    */
/*This program CONVERTS a user specified 400X512 TARGA file to TIFF.  */
/*A 222 byte header is added to the TARGA data to create this file    */
/*which imported into Word Perfect 5.0.  The 222 byte header is read  */
/*from the file "tifhead.fil" which must be present for this program  */
/*to work.  Since TARGA files start from the bottom and this header   */
/*specifies starting from the top, inversion is necessary.  The       */
/*program TAR2TIF2.C can convert TARGA files of any size, but runs    */
/*slower.                                                             */
/**********************************************************************/

#include "stdio.h"
#include "string.h"

int i,j,k;
char holder[512], buffer[100][512];
char filename[256], filename2[256];
char tifhead[222];

FILE *file1, *file2;

void main()
{
   /*read in header*/
   file1=fopen("tifhead.fil","rb");
   fread(tifhead,1,222,file1);
   fclose(file1);

   /*get file names and open files*/
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of TIFF file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*write header*/
   fwrite(tifhead,1,222,file2);

   /*read in TARGA, invert, and write TIFF*/
   printf("Count to 4\n");
   for(i=0; i<4; i++)
   {
      printf("%d\n",i+1);
      fseek(file1, 18+153600-i*51200,0);
      fread(buffer, 1, 51200, file1);
      for(j=0; j<50; j++)
      {
         for(k=0; k<512; k++)
         {
            holder[k]=buffer[j][k];
            buffer[j][k]=buffer[99-j][k];
            buffer[99-j][k]=holder[k];
         }
      }
      fwrite(buffer, 51200, 1, file2);
   }
   fclose(file1);
   fclose(file2);
}
/*********************************************************************/
```

```
/**********************************************************************/
/* TAR2TIF2.C                                                         */
/*                                                                    */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*  CONVERSION PROGRAM FOR TARGA TO TAGGED IMAGE FILE FORMAT (TIFF)   */
/*                       by Capt John Cline                           */
/*                        October 31, 1989                            */
/*                                                                    */
/*This program CONVERTS a user specified TARGA file of any size to    */
/*TIFF.  A 222 byte header is added to the TARGA data to create this  */
/*file which imported into Word Perfect 5.0.  The 222 byte header is  */
/*read from the file "tifhead.fil" which must be present for this     */
/*program to work.  The header must be modified to reflect the proper*/
/*file size.  Also, since TARGA files start from the bottom and this  */
/*header specifies starting from the top, inversion is necessary.     */
/*The program TAR2TIF.C can convert only 400X512 TARGA files, but is  */
/*much faster.                                                        */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
long int seek_length;
int i, columns, rows;
unsigned char filename[256], filename2[256], buffer[512], targhead[18];
unsigned char tifhead[222];

FILE *file1, *file2;

void main()
{
   /*read in header*/
   file1=fopen("tifhead.fil","rb");
   fread(tifhead,1,222,file1);
   fclose(file1);

   /*get file names and open files*/
   printf("Enter complete path and file name of TARGA file.\n");
   scanf("%s", filename);
   file1=fopen(filename,"rb");
   printf("Enter complete path and file name of TIFF file.\n");
   scanf("%s", filename2);
   file2=fopen(filename2,"wb");

   /*get TARGA input file specs and write Imaging Technology header*/
   fread(targhead, 1, 18, file1);
   columns=(int)targhead[12]+256*(int)targhead[13];
   rows=(int)targhead[14]+256*(int)targhead[15];
   tifhead[30]=targhead[12];
   tifhead[31]=targhead[13];
   tifhead[42]=targhead[14];
   tifhead[43]=targhead[15];
   fwrite(tifhead,1,222,file2);
   /*read in TARGA and write TIFF one row at a time*/
   for(i=rows-1; i>=0; i--)
   {
      seek_length = (long)i * (long)columns + 18;
      fseek(file1, seek_length, 0);
      fread(buffer, 1, columns, file1);
      fwrite(buffer, columns, 1, file2);
   }
   fclose(file1);
   fclose(file2);
   printf("done \n");
}
/***********************************************************************/
```

# APPENDIX G. EQUIPMENT CONTROL SOFTWARE

This appendix presents information relating to software written for equipment control during this thesis. The equipment controlled by this software are the MOSLM and the TARGA framegrabber, discussed in Appendices A and B respectively. The software includes DOS batch files, .CMD files for use with the vendor program TESTTARG.EXE (discussed in Appendix B), and programs written in "C" for Borland's Turbo-C compiler.

Much of the code assumes certain locations for various files. The program TESTTARG.EXE, and its associated .CMD files, as well as all TARGA image files are assumed to be in the \TARGA directory. The compiled versions (.EXE files) of the Turbo-C programs are assumed to be in the \TURC directory. The Turbo-C programs also occaisionally call on support files that must also be located in the \TURC directory. The batch files were located in the root directory of the hard drive, and the various other files were properly placed at the time of writing this thesis.

This appendix is divided into three sections which relate to alignment software, JTC software, and LPCT software. All software will appear at the end of the section. Presentation order will follow the order of application discussion. Batch files will be presented first, followed by the .CMD files used with TESTTARG.EXE, then the Turbo-C source code.

## G.1 Alignment Software

Six useful MOSLM patterns were available to assist in alignment of the hybrid optical/digital architecture. The test signal, two checkerboard patterns, and the crosshair were described in Appendix A, and the software needed to display these patterns was also presented. Two other patterns were also useful; these were three annuli displayed simultaneously, and the wagon wheel test pattern pictured in Figure 7a of Chapter III and used in the experiments of Chapter IV.

The first three patterns required only a live camera to be useful (see LIVE.BAT in Appendix B). However, the crosshair pattern could either be helpful with the camera in live mode, or when used with the TESTTARG.EXE command file "x-hair.cmd".

To ensure proper capture of Fourier features (step 3 of Figure 29 in Chapter V), CCD1 (see Figure 28) had to be positioned so that the origin of Fourier plane was centered within the capture region of the TARGA framegrabber. By running the X-HAIR.C (listed in Appendix A), a crosshair pattern was seen on CCD1. The batch file X-HAIR.BAT using TESTTARG.EXE with the command file "x-hair.cmd", would capture a centered 128X128 region and redisplay it with a black crosshair superimposed on it. CCD1 could then be aligned so that the black crosshair lined up with the bright one created by the MOSLM pattern.

The three annuli program, 3-ANNULI.C, displayed annuli in each of the top two corners of the MOSLM, and a third in centered in the lower half. When the resultant pattern was viewed in the Fourier plane, interference fringes could be seen from the three combinations of annuli pairs. This pattern could be used to test for proper focus in the Fourier plane.

The last MOSLM pattern program, W-WHEEL.C, was used to test the LPCT portion of the hybrid architecture as discussed in Chapter IV. Using it, and the TESTTARG.EXE command file "110X456.cmd", CCD2 of Figure 28 could be properly aligned, since "110X456.cmd" was also used during execution of the batches associated with the hybrid architecture. The DOS batch file 110X456.BAT would properly execute TESTTARG.EXE, after running W-WHEEL.C.

```
:X-HAIR.BAT
:This batch file will execute TESTTARG using the command file
:"x-hair.cmd" after initializing the TARGA variables.  It is
:useful for centering a camera, and should be run after
:running X-HAIR.C.

echo off
cls
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <x-hair.cmd
```

```
:X-HAIR.CMD
:This is a command file to be executed with the program TESTTARG,
:using the DOS batch file command C>X-HAIR.  It will grab the
:center 128X128 pixels on the live TARGA monitor.  Then redisplay
:this frame after drawing a dark crosshair through the center.
:It is useful for centering a camera, and should be used after
:running the program X-HAIR.C.

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER CAPTURE
erase
0
0
0
erase
0
0
0

:GRAB FRAME
grab
dis

:DRAW CROSSHAIR
line
2
255
0
255
399
-1
line
2
0
199
511
199
-1
```

```
:SAVE CENTER 128X128
putpic
grab128
192
136
319
263
-1
erase
0
0
0

:REDISPLAY CENTER 128X128
getpic
grab128
0
0
-1
graphend
quit
```

```
/**********************************************************************/
/* 3-ANNULI.C                                                         */
/*                                                                    */
/*              AIR FORCE INSTITUTE OF TECHNOLOGY                     */
/*                CAMERA ALIGNMENT TEST PATTERN                       */
/*                     by Capt John Cline                             */
/*                       July 31, 1989                                */
/*                                                                    */
/*This program creates a test pattern containing three annuli, one    */
/*in each of the two top corners of the SLM, and one centered on the */
/*lower half of the SLM.  When the camera is properly focused in the */
/*Fourier plane, fringes associated with the three pairs of annuli   */
/*will be visible.                                                    */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, x, y, xc, yc, col, row;
float outradsquared=196, inradsquared=100, test;
float xref=64, yref=14;
unsigned char slm[128][64], temp[128][128], filename[256];

void main()
{
   prepare_slm();

   /*initialize temp matrix*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         temp[row][col]=1;
      }
   }

   /*write the reference circle*/
   for (col=50; col<79; col++)
   {
      for(row=0; row<29; row++)
      {
         test=(row-yref)*(row-yref)+(col-xref)*(col-xref);
         if(test<=outradsquared && test>=inradsquared)
            temp[row][col]=0;
      }
   }

   /*write the top circles*/
   for (row=99; row<128; row++)
   {
      for(col=0; col<29; col++)
      {
         test=(row-113)*(row-113)+(col-14)*(col-14);
         if(test<=outradsquared && test>=inradsquared)
            temp[row][col]=0;
      }
      for(col=99; col<128; col++)
      {
         test=(row-113)*(row-113)+(col-113)*(col-113);
         if(test<=outradsquared && test>=inradsquared)
            temp[row][col]=0;
      }
   }

   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                       /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/**********************************************************************/
```

```
/**********************************************************************/
/* W-WHEEL.C                                                          */
/*                                                                    */
/*               AIR FORCE INSTITUTE OF TECHNOLOGY                    */
/*           LOG POLAR COORDINATE TRANSFORM TEST PATTERN              */
/*                     by Capt John Cline                             */
/*                      August 31, 1989                               */
/*                                                                    */
/*This program creates a test pattern containing three concentric    */
/*circles, whose inner and outer radii are logarithmic related, and  */
/*eight radial lines, every 45 degrees apart.  The pattern is used to*/
/*test the LPCT CGH, since the result coordinate transform is an     */
/*easily predictable grid pattern.                                    */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"
#include "time.h"

int i, j, k, x, y, col, row, inner, outer;
float outer_squared, inner_squared, test;
unsigned char slm[128][64], temp[128][128];

void main()
{
   prepare_slm();

   /*initialize temp matrix*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         temp[row][col]=1;
      }
   }

   printf("calculating\n");

   /*create + */
   for(i=0; i<128; i++)
   {
      for(j=62; j<66; j++)
      {
         temp[i][j]=0;
         temp[j][i]=0;
      }
   }

   /*create X*/
   for(i=18; i<110; i++)
   {
      for(j=18; j<110; j++)
      {
         if(abs(i-j)<=2 || abs(i+j-127)<=2) temp[i][j]=0;
      }
   }

   /*create circles*/
   for(inner=15; inner<64; inner=inner*2)
   {
      inner_squared=inner*inner;
      outer=inner*16/15;
      outer_squared=outer*outer;
      for (col=64-outer; col<65+outer; col++)
      {
         for(row=64-outer; row<65+outer; row++)
         {
            test=(row-64)*(row-64)+(col-64)*(col-64);
            if(test<=outer_squared && test>=inner_squared)
               temp[row][col]=0;
         }
      }
   }

   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                 /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
  }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
```

```
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*******************************************************************/
```

```
:110X456.BAT
:This batch file is used to grab a 110X456 pixel region on the TARGA
:framegrabber system.  This region represents the area of interest for
:the log-polar coordinate transformation.  By using this batch file,
:which executes TESTTARG.EXE using the command file "110X456.cmd",
:proper alignment of the camera can be achieved without performing the
:entire process.  The test pattern generated by W-WHEEL.C can provide
:an appropriate pattern for alignment, so that program should be run
:before executing this batch.

echo off
cls
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <110x456.cmd



:110x456.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used at various stages of the log polar coordinate transform
:routines.  After grabbing a region of 110X456 pixels, the data is
:in TARGA format in the file "grablrt.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE PROPER 110x456
putpic
grablrt
28
48
483
158
-1
erase
0
0
0
```

```
:REDISPLAY SAVED SECTION
getpic
grablrt
0
0
-1
graphend
quit
```

## G.2 JTC Software

There were five ways to execute the JTC for this thesis: 1) the mean difference method of fringe binarization; 2) using a simple mean value threshold for fringe binarization; 3) the median value threshold; 4) the zero difference method; and 5) a segmented scene input using the zero difference method. These correspond to Figures 14 - 17 and 20 of Chapter III, respectively, and their associated batch files are listed as JTHF-1.BAT through JTHF-5.BAT.

Two other types of JTC batch files were also possible: JTC-CAL.BAT and JTC-TST.BAT. These two batches provided calibration for the JTC and tested that calibration using two small annuli as scene and template inputs. The technique of JTHF-1.BAT was used. JTHF-CAL.BAT had to be run four times to complete the calibration process. Each time the program was run the user would have to select a different corner to display the scene annulus in. After all four corners were finished, a complete set of data was available for linear interpolation of correlation planes. JTHF-TST.BAT could then be run to verify the calibration, since it too displayed annuli, at a user specified location.

All seven of the batch files will be presented with JTHF-CAL.BAT first, then JTHF-TST.BAT next, followed by the other five presented in order. After the batch files, the .CMD files, used with TESTTARG.EXE, will be presented in their order of appearance. Finally, the Turbo-C source code will be presented, again in its order of appearance.

```
:JTHF-CAL.BAT
:This is the batch file to calibrate the Joint Transform
:Correlator using the average difference method.  It
:must be run four times to complete calibration.

echo off
cls

:display scene and template annuli
cd \turc
call-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:display scene annulus alone
cd \turc
disptop

:capture Fourier transform of scene alone (256X256 pixels)
CD \TARGA
testtarg <top-hf.cmd
cls

:reduce the two 256X256 files to 128X128
cd \turc
tar2hlf

:Find the pixel by pixel difference and binarize on average
:difference, then display on SLM.
jt-diff

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and save as cal data
cal2-hf
```

```
:JTHF-TST.BAT
:This is the batch file for testing the calibration of
:the Joint Transform Correlatorusing the average
:difference method.  The only difference between this and
:JTHF-1.BAT, is the first Turbo-c program executed.

echo off
cls

:display scene and template annuli
cd \turc
circ-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:display scene alone
cd \turc
disptop

:capture Fourier transform of scene alone (256X256 pixels)
CD \TARGA
testtarg <top-hf.cmd
cls

:reduce the two 256X256 files to 128X128
cd \turc
tar2hlf

:Find the pixel by pixel difference and binarize on average
:difference, then display on SLM.
jt-diff

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JTHF-1.BAT
:This is the batch file for the Joint Transform Correlator
:using the average difference method.

echo off
cls

:display scene and template
cd \turc
dis-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:display scene alone
cd \turc
disptop

:capture Fourier transform of scene alone (256X256 pixels)
CD \TARGA
testtarg <top-hf.cmd
cls

:reduce the two 256X256 files to 128X128
cd \turc
tar2hlf

:Find the pixel by pixel difference and binarize on average
:difference, then display on SLM.
jt-diff

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JTHF-2.BAT
:This is the batch file for the Joint Transform Correlator
:using the mean value fringe binarization technique.

echo off
cls

:display scene and template
cd \turc
dis-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:Reduce 256X256 to 128X128, binarize on mean value, and
:display on the SLM.
cd \turc
ft_both

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JTHF-3.BAT
:This is the batch file for the Joint Transform Correlator
:using the median value fringe binarization technique.

echo off
cls

:display scene and template
cd \turc
dis-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:Reduce 256X256 to 128X128, binarize on median value, and
:display on the SLM.
cd \turc
jt2med

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JTHF-4.BAT
:This is the batch file for the Joint Transform Correlator
:using the zero difference method.

echo off
cls

:display scene and template
cd \turc
dis-hf

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jtl-hf.cmd
cls

:display scene alone
cd \turc
disptop

:capture Fourier transform of scene alone (256X256 pixels)
CD \TARGA
testtarg <top-hf.cmd
cls

:reduce the two 256X256 files to 128X128
cd \turc
tar2hlf

:Find the pixel by pixel difference and binarize on zero
:difference (i.e. sign of difference), then display on SLM.
jt_comb

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JTHF-5.BAT
:This is the batch file for the Joint Transform Correlator
:using the zero difference method.  This batch differs from :JTHF-4.BAT
in that it performs the JTC using the segmented :scene from the previous
pass through the JTC.

echo off
cls

:display segmented scene and template
cd \turc
dis-hf2

:capture the joint transform (256X256 pixels)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <jt1-hf.cmd
cls

:display segmented scene alone
cd \turc
disptop2

:capture Fourier transform of scene alone (256X256 pixels)
CD \TARGA
testtarg <top-hf.cmd
cls

:reduce the two 256X256 files to 128X128
cd \turc
tar2hlf

:Find the pixel by pixel difference and binarize on zero
:difference (i.e. sign of difference), then display on SLM.
jt_comb

:Capture correlation plane (320X256 pixels)
CD \TARGA
testtarg <jt2-hf.cmd
cls

:Reduce 320X256 to 160X128
cd \turc
tar2hlf2

:Locate peaks and interpret
bull-hf2

:Display results on TARGA monitor
CD \TARGA
testtarg <jt3.cmd
```

```
:JT1-HF.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 256X256
:pixel region.  The data is stored in the file "grabl_2.tga", and will
:be later reduced to 128X128 by the program TAR2HLF.C.

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 256X256
putpic
grabl_2
128
72
383
327
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
grabl_2
0
0
-1
graphend
quit
```

```
:TOP-HF.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 256X256
:pixel region.  The data is stored in the file "top_2.tga", and will
:be later reduced to 128X128 by the program TAR2HLF.C.

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 256X256
putpic
top_2
128
72
383
327
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
top_2
0
0
-1
graphend
quit
```

```
:JT2-HF.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 320X256
:pixel region.  The data is stored in the file "grab2_2.tga", and will
:be later reduced to 160X128 by the program TAR2HLF2.C.

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 320X256
putpic
grab2_2
128
40
383
359
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
grab2_2
0
0
-1
graphend
quit
```

```
:JT3.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used to display the results of the joint transform correlator.

live
erase
0
0
0
erase
0
0
0
dis

:GET ORIGINAL INPUT
getpic
display2
0
271
-1

:GET JOINT TRANSFORM
getpic
grab1
128
271
-1

:GET FOURIER OF SCENE ALONE
getpic
top
256
271
-1

:GET BINARY VERSION OF FRINGES
getpic
binary
384
271
-1

:GET CORRELATION PLANE
getpic
grab2
0
100
-1
```

```
:GET CORRELATION PLANE WITH LOCATED PEAKS
getpic
max
128
100
-1

:GET ORIGINAL SCENE WITH TARGETS MARKED WITH X's
getpic
bullseye
256
100
-1

:GET SEGMENTED SCENE
getpic
segment
384
100
-1

:SAVE ENTIRE DISPLAY
putpic
grab3
0
100
511
399
-1
graphend
quit
```

```
/**************************************************************************/
/* CAL1-HF.C                                                             */
/*                                                                       */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*  INPUT DISPLAY FOR CALIBRATION OF THE JOINT TRANSFORM CORRELATOR      */
/*                        by Capt John Cline                             */
/*                          August 31, 1989                              */
/*                                                                       */
/*This program displays a target circle in one of four user specified*/
/*corners in the top half, and a reference circle in the bottom half */
/*of the SLM.  It modifies the file "cal.dat" to let the program    */
/*CAL2-HF.C know which corner is being used.  It also creates a TARGA*/
/*file in "display2.tga" for viewing on the TARGA screen.  This      */
/*program is the first of a series that will provide calibration to */
/*the joint transform correlator.                                    */
/**************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, x, y, xc, yc, col, row, corner;
float outradsquared=16, inradsquared=4, test;
float xref=64, yref=103;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];
unsigned char caldata[17];

FILE *file1;

void main()
{
   prepare_slm();

   /*get cal data file*/
   file1=fopen("\\turc\\cal.dat","rb");
   fread(caldata,1,17,file1);
   fclose(file1);

   /*find out which corner to change*/
   printf("Enter number of corner to change:\n");
   printf("    1) 7-34\n");
   printf("    2) 120-34\n");
   printf("    3) 7-64\n");
   printf("    4) 120-64\n");
   scanf("%d",&corner);
   caldata[16]=(unsigned char)(corner-1);

   /*assign the center values*/
   xc=7;
   yc=34;
```

```
    if(corner==2) xc=120;
    if(corner==3) yc=64;
    if(corner==4) xc=120;
    if(corner==4) yc=64;

    /*write the updated file*/
    file1=fopen("\\turc\\cal.dat","wb");
    fwrite(caldata, 17, 1, file1);
    fclose(file1);

    printf("calculating\n");

    /*write the reference circle*/
    for (col=60; col<69; col++)
    {
       for(row=99; row<108; row++)
       {
          test=(row-yref)*(row-yref)+(col-xref)*(col-xref);
          if(test<=outradsquared && test>=inradsquared)
             targabuffer[127-row][col]=255;
       }
    }

    /*write the target circle*/
    for (col=xc-4; col<xc+5; col++)
    {
       for(row=yc-4; row<yc+5; row++)
       {
          test=(row-yc)*(row-yc)+(col-xc)*(col-xc);
          if(test<=outradsquared && test>=inradsquared)
             targabuffer[127-row][col]=255;
       }
    }

    save_targabuffer_to_display2();
    convert_targabuffer_to_temp();

    convert_temp_to_slm();
    write_to_slm();
}

/**  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
    char far *semetex;
    semetex = (char far *) 0xb0000000;
    *(semetex+0x800)=1;                                  /*clear slm*/
}

/**  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
save_targabuffer_to_display2()
{
    /*save TARGA file*/
```

```
   printf("writing TARGA file to targa\\display2.tga\n");
   filel=fopen("\\targa\\display2.tga","wb");
   fwrite(targhead, 18, 1, filel);
   fwrite(targabuffer, 16384, 1, filel);
   fclose(filel);
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+[illegible];
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+2+3;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+20;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+64;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
```

```
          *(semetex+i) = slm[x][3+4*j];
        }
    }
    *(semetex+0x801)=1;
}
/*****************************************************************************/
```

```
/***************************************************************************/
/* DISPTOP.C                                                              */
/*                                                                        */
/*                    AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*          SCENE ONLY DISPLAY TO THE JOINT TRANSFORM CORRELATOR          */
/*                          by Capt John Cline                            */
/*                            August 31, 1989                             */
/*                                                                        */
/*This program displays only the scene half (upper 71x128 pixels) of     */
/*the TARGA file "display2.tga" onto the SLM.  This program is part      */
/*of the joint transform correlator, and is run after the scene and      */
/*template have been displayed together.                                 */
/***************************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];

FILE *file1;

void main()
{
   prepare_slm();

   /*read in scene half only*/
   file1=fopen("\\targa\\display2.tga","rb");
   fseek(file1,7314,0);
   fread(&targabuffer[57][0],1,9088,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                      /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/**********************************************************************
```

```
/**************************************************************************/
/* TAR2HLF.C                                                              */
/*                                                                        */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                    */
/* PROGRAM FOR REDUCING FILES USED IN THE JOINT TRANSFORM CORRELATOR      */
/*                        by Capt John Cline                              */
/*                         August 31, 1989                                */
/*                                                                        */
/*This program REDUCES two TARGA files "grabl_2.tga" and "top_2.tga"      */
/*from 256X256 to 128X128 and stores them in the TARGA files              */
/*"grabl.tga" and "top.tga" respectively.  The reduction is               */
/*accomplished by averaging.  This program is to be used in the           */
/*joint transform correlator.                                             */
/**************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k;
unsigned char bufferin[128][256];
unsigned char bufferout[64][128];
unsigned holder;
char targhead[]   {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   printf("reducing files\n\n");

   /*open first pair of files*/
   file1=fopen("\\targa\\grabl_2.tga","rb");
   file2=fopen("\\targa\\grabl.tga","wb");

   /*handle headers*/
   fseek(file1, 18, 0);
   fwrite(targhead, 18, 1, file2);

   /*read in data, reduce, and write it out*/
   for(i=0; i<2; i++)
   {
      fread(bufferin, 1, 32768, file1);
      for(j=0; j<128; j+=2)
      {
         for(k=0; k<256; k+=2)
         {
            holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]+
               bufferin[j+1][k+1])/4;
            bufferout[j/2][k/2]=holder;
         }
      }
      fwrite(bufferout, 8192, 1, file2);
   }
   fclose(file1);
   fclose(file2);


   /*open second set of files*/
   file1=fopen("\\targa\\top_2.tga","rb");
   file2=fopen("\\targa\\top.tga","wb");

   /*handle headers*/
   fseek(file1, 18, 0);
   fwrite(targhead, 18, 1, file2);

   /*read data, reduce, and write*/
   for(i=0; i<2; i++)
   {
      fread(bufferin, 1, 32768, file1);
      for(j=0; j<128; j+=2)
      {
         for(k=0; k<256; k+=2)
         {
            holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
               +bufferin[j+1][k+1]+2)/4;
            bufferout[j/2][k/2]=holder;
         }
      }
      fwrite(bufferout, 8192, 1, file2);
   }
   fclose(file1);
   fclose(file2);
}
/**********************************************************************/
```

```
/*************************************************************************/
/* JT-DIFF.C                                                             */
/*                                                                       */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*  PROGRAM FOR BINARIZING FRINGES IN THE JOINT TRANSFORM CORRELATOR    */
/*                        by Capt John Cline                             */
/*                         August 31, 1989                               */
/*                                                                       */
/*This program binarizes the fringes of the JTC by finding the mean     */
/*difference between the joint transform file "grabl.tga" and the       */
/*Fourier transform of the scene file "top.tga", and using it as a      */
/*threshold.  If the difference is greater than the threshold the       */
/*pixel would be on, if less, the pixel is off.  The TARGA file         */
/*"binary.tga" is saved with the binary version of the fringes, and     */
/*the SLM is appropriately written to for the next step of the JTC.     */
/*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

unsigned char targhead[18], slm[128][64];
unsigned char joint[128][128], targabuffer[128][128], temp[128][128];
int maxvalue, i, j, x, y, col, row, index;
float combined, thrshld=0;

FILE *filel;

void main()
{
   prepare_slm();

   /*get joint transform file*/
   filel=fopen("\\targa\\grabl.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                          /*read header*/
   fread(joint,1,16384,filel);
   fclose(filel);

   /*get scene only file*/
   filel=fopen("\\targa\\top.tga","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*calculate threshold by finding average difference*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         thrshld+=joint[row][col]-targabuffer[row][col];
      }
   }
   thrshld=thrshld/16384+2;

   /*binarize pixels based on threshold for difference*/
   printf("binarizing data \n");
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         combined=joint[row][col]-targabuffer[row][col];
         if(combined<=thrshld)
         {
            targabuffer[row][col]=0x00;
         }
         else
         {
            targabuffer[row][col]=0xff;
         }
      }
   }

   /*save binarized version */
   file1=fopen("\\targa\\binary.tga","wb");
   fwrite(targhead, 18, 1, file1);
   fwrite(targabuffer, 16384, 1, file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                    /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
```

```
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128: x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]   =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*****************************************************************************/
```

```
/**********************************************************************/
/* TAR2HLF2.C                                                         */
/*                                                                    */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                */
/* PROGRAM FOR REDUCING FILES USED IN THE JOINT TRANSFORM CORRELATOR */
/*                        by Capt John Cline                          */
/*                         August 31, 1989                            */
/*                                                                    */
/*This program REDUCES the TARGA files "grab2_2.tga" from 320X256 to */
/*160X128 and stores it in the TARGA file "grab2.tga".  The reduction*/
/*is accomplished by averaging.  This program is to be used in the   */
/*joint transform correlator.                                        */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k;
unsigned char bufferin[160][256];
unsigned char bufferout[80][128];
unsigned holder;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00'
                   '\x80', '\x00', '\xa0', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   printf("\n\nreducing files\n\n");
   /*open files*/
   file1=fopen("\\targa\\grab2_2.tga","rb");
   file2=fopen("\\targa\\grab2.tga","wb");

   /*handle headers*/
   fseek(file1, 18, 0);
   fwrite(targhead, 18, 1, file2);


   /*read data in, reduce, and write*/
   for(i=0; i<2; i++)
   {
      fread(bufferin, 1, 40960, file1);
      for(j=0; j<160; j+=2)
      {
         for(k=0; k<256; k+=2)
         {
            holder=(bufferin[j][k]+bufferin[j+1][k]+bufferin[j][k+1]
               +bufferin[j+1][k+1]+2)/4;
            bufferout[j/2][k/2]=holder;
         }
      }
```

```
        fwrite(bufferout, 10240, 1, file2);
    }
    fclose(file1);
    fclose(file2);
}
/*********************************************************************/
```

```
/***********************************************************************/
/* CAL2-HF.C                                                           */
/*                                                                     */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/*   PEAK LOCATOR FOR CALIBRATING THE JOINT TRANSFORM CORRELATOR       */
/*                       by Capt John Cline                            */
/*                         August 31, 1989                             */
/*                                                                     */
/*This program finds two maximum values, a top half value and a       */
/*bottom half value, and their coordinates in the TARGA file          */
/*"grab2.tga".  It then modifies the calibration file "cal.dat" to    */
/*include these new values.  The program BULL-HF.C uses "cal.dat" to  */
/*interpret results from the joint transform correlator.  By relating*/
/*scene annuli locations, known apriori and created with the program */
/*CAL1-HF.C, to their corresponding correlation peak locations in the*/
/*file "grab2.tga" created by running the joint transform correlator,*/
/*the JTC can be calibrated to properly interpret target locations   */
/*within arbitrary scenes.                                            */
/***********************************************************************/

#include "stdio.h"
#include "string.h"

int i,j;
unsigned char targabuffer[160][128];
unsigned char caldata[4][4],corner[1];
unsigned int topx=0, topy=0, topvalue=0, corn;
unsigned int botx=0, boty=0, botvalue=0;

FILE *file1;

main()
{
   printf("reading in data \n");
   /*get ccd file*/
   file1=fopen("\\targa\\grab2.tga","rb");
   fseek(file1,18,0);
   fread(targabuffer,1,20480,file1);
   fclose(file1);

   /*get cal data file*/
   file1=fopen("\\turc\\cal.dat","rb");
   fread(caldata,1,16,file1);
   fread(corner,1,1,file1);
   fclose(file1);

   printf("%d\n",corner[0]+1);
   for(i=0;i<4;i++)
   {
      printf("%d %d ",caldata[i][0], caldata[i][1]);
      printf("%d %d\n",caldata[i][2],caldata[i][3]);

   }
```

```
/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/

   /*locate max value on the bottom*/
   for(i=0; i<45; i++)
   {
      for(j=0; j<128; j++)
      {
         if (targabuffer[i][j] > botvalue)
         {
            botvalue=targabuffer[i][j];
            botx=j;
            boty=i;
         }
      }
   }

   /*locate the max value on the top*/
   for(i=115; i<160; i++)
   {
      for(j=0; j<128; j++)
      {
         if (targabuffer[i][j] > topvalue)
         {
            topvalue=targabuffer[i][j];
            topx=j;
            topy=i;
         }
      }
   }
/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/

   corn=(unsigned int)corner[0];
   caldata[corn][0]=(unsigned char)topx;
   caldata[corn][1]=(unsigned char)topy;
   caldata[corn][2]=(unsigned char)botx;
   caldata[corn][3]=(unsigned char)boty;

   if(topx==0 && topy==130) printf("\nRUN THIS ONE AGAIN!!\n\n");
   if(topx<=73 && topx>=53) printf("\nCHECK ALIGNMENT!!\n\n");
   if(botx<=73 && botx>=53) printf("\nCHECK ALIGNMENT!!\n\n");

   printf("%u\n",corner[0]);
   for(i=0;i<4;i++)
   {
      printf("%d %d ",caldata[i][0], caldata[i][1]);
      printf("%d %d\n",caldata[i][2], caldata[i][3]);
   }

   file1=fopen("\\ture\\cal.dat","wb");
   fwrite(caldata, 16, 1, file1);
   fwrite(corner, 1, 1, file1);
   fclose(file1);
   printf("press any key to continue");
   getch();
}
/*******************************************************************************
```

```
/*************************************************************************
/* CIRC-HF.C
 *
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY
/*       INPUT DISPLAY FOR TESTING THE JOINT TRANSFORM CORRELATOR
/*                        by Capt John Cline
/*                          August 21, 1989
/*
/*This program creates an array with a target circle of user
/*specified location in the top half, and a reference circle in the
/*bottom half.  It also creates a TARGA file in "display.tga" for
/*viewing on the TARGA screen.  Finally, it displays the array on
/*the SLM.  It will be used for testing the calibration of the joint
/*transform correlator.
*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, x, y, xc, yc, col, row;
float outradsquared=16, inradsquared=4, test;
float xref=64, yref=103;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', [illegible]
                   '\x00', '\x00', '\x00', '\x00', [illegible]
                   '\x80', '\x00', '\x80', [illegible]
unsigned char slm[128][64], targabuffer[128][128], temp[illegible]

FILE *file1;

void main()
{
   prepare_slm();

   /*get circle coordinates and open file*/
   printf("Enter  circle  center  coordinates,  xc  between [illegible]
inclusive");
   printf("\nand yc between 34 and 64 inclusive.\n");
   printf("Enter xc (must be an integer).\n");
   scanf("%d",&xc);
   printf("Enter yc (must be an integer):\n");
   scanf("%d",&yc);

   printf("calculating\n");

   /*write the reference circle*/
   for (col=60; col<69; col++)
   {
      for(row=99; row<108; row++)
      {
         test=(row-yref)*(row-yref)+(col-xref)*(col-xref);
         if(test<=outradsquared && test>=inradsquared)
            targabuffer[127-row][col]=255;
      }
   }

   /*write the target circle*/
   for (col=xc-4; col<xc+5; col++)
   {
      for(row=yc-4; row<yc+5; row++)
      {
         test=(row-yc)*(row-yc)+(col-xc)*(col-xc);
         if((test<=outradsquared && test>=inradsquared))
            targabuffer[127-row][col]=255;
      }


   save_targabuffer_to_display2();
   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();


/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()

   char far *semetex;
   semetex = (char far *) 0xda000000;
   *(semetex+0x8000)=1;                                    /*clear*/


/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
save_targabuffer_to_display2()

   /*save TARGA file*/
   printf("writing TARGA file to targa\\display2.tga\n");
   file1=fopen("\\targa\\display2.tga","wb");
   fwrite(targhead, 18, 1, file1);
   fwrite(targabuffer, 16384, 1, file1);
   fclose(file1);
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0xff;
         else temp[i][j]=0x00;

   }


/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
```

```
convert_temp_to_slm()
{
   /*convert 8 pixels into an eight bit character for addr[illegible]

   printf("creating SLM format \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j] =2*temp[126-8*j][x]+temp[127-8*j][x];
         slm[x][4*j+1] 8*temp[124-8*j][x]+4*temp[125-8*j][x];
         slm[x][4*j+2] 32*temp[122-8*j][x]+16*temp[123-8*j][x];
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x];
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** */
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/***********************************************************************/
```

```
/*************************************************************************/
/* BULL-HF.C                                                             */
/*                                                                       */
/*                   AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*  CORRELATION PLANE INTERPRETER FOR THE JOINT TRANSFORM CORRELATOR    */
/*                          by Capt John Cline                           */
/*                           August 31, 1989                             */
/*                                                                       */
/*This program finds the top 10 values in the top and bottom cross-     */
/*correlation regions of the joint transform correlator correlation     */
/*plane file "grab2.tga".  It rewrites the correlation plane file       */
/*with crosshairs at peak locations in the cross correlation regions    */
/*to the file "max.tga".  Next, it uses these max values, along with    */
/*earlier calibration values stored in "cal.dat", to calculate the      */
/*coordinates of the target in the original scene.  (This data is      */
/*displayed on the computer monitor.)  Having done this, the original  */
/*scene file ("display2.tga) is rewritten with crosshairs at the        */
/*target location as "bullseye.tga".                                    */
/*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, toptest=0, bottest=0, xb[10], yb[10], xt[10], yt[10];
int ii, jj, ilow, ihigh, jlow, jhigh;
unsigned char targaheader[18], targa[160][128], outbuffer[128][128];
unsigned int topx[11], topy[11], topvalue[11];
unsigned int botx[11], boty[11], botvalue[11];
unsigned char caldata[4][4];
float cal[4][4], xx, yy;
float x1, x2, y1, y2, xfb[10], xft[10], yfb[10], yft[10];

FILE *file1;

void main()
{
   /*initialize values*/
   for(i=0; i<11; i++)
   {
      topx[i]=0;
      topy[i]=0;
      topvalue[i]=0;
      botx[i]=0;
      boty[i]=0;
      botvalue[i]=0;
   }

   /*get ccd file*/
   file1=fopen("\\targa\\grab2.tga","rb");
   fread(targaheader,1,18,file1);
   printf("\n\nreading in data \n");
   fread(targa,1,20480,file1);
```

```
fclose(file1);

/*get cal data file*/
file1=fopen("\\turc\\cal.dat","rb");
fread(caldata,1,16,file1);
fclose(file1);
for(i=0; i<4; i++)
{
   for(j=0; j<4; j++)
   {
      cal[i][j]=(float)caldata[i][j];
   }
}

/*erase center autocorrelation section*/
for(i=45; i<115; i++)
{
   for(j=0; j<128; j++)
   {
      targa[i][j]=0x00;
   }
}

/*locate the 10 max values on the top*/
for(i=115; i<160; i++)
{
   for(j=0; j<128; j++)
   {
      if(targa[i][j]>=100)
      {
         if(toptest<10) toptest++;
         k=9;
         while(targa[i][j] >topvalue[k] && k>=0)
         {
            topvalue[k+1]=topvalue[k];
            topx[k+1]=topx[k];
            topy[k+1]=topy[k];
            topvalue[k]=targa[i][j];
            topx[k]=j;
            topy[k]=i;
            k--;
         }
      }
   }
}

/*locate 10 max values on the bottom*/
for(i=0; i<45; i++)
{
   for(j=0; j<128; j++)
   {
      if(targa[i][j]>=100)
      {
            if(bottest<10) bottest++;
            k=9;
            while(targa[i][j] >botvalue[k] && k>=0)
            {
               botvalue[k+1]=botvalue[k];
               botx[k+1]=botx[k];
               boty[k+1]=boty[k];
               botvalue[k]=targa[i][j];
               botx[k]=j;
               boty[k]=i;
               k--;
            }
         }
      }
   }

   /*write file with crosshairs*/
   for(j=0; j<bottest; j++)
   {
      for(i=-3; i<4; i++)
      {
         targa[boty[j]+i][botx[j]]=0x00;
         targa[boty[j]][botx[j]+i]=0x00;
      }
   }
   for(j=0; j<toptest; j++)
   {
      for(i=-3; i<4; i++)
      {
         targa[topy[j]+i][topx[j]]=0x00;
         targa[topy[j]][topx[j]+i]=0x00;
      }
   }

   printf("writing correlation file\n");
   file1=fopen("\\targa\\max.tga","wb");
   fwrite(targaheader,18,1,file1);
   fwrite(targa,20480,1,file1);
   fclose(file1);

/*This part of the program finds the location of the target given the*/
/*location of the correlation peak.  It puts cross hairs on the      */
/*original scene file.                                               */

   /*read in the original scene file*/
   printf("reading in original scene file\n");
   file1=fopen("\\targa\\display2.tga","rb");
   fread(targaheader, 1, 18, file1);
   fread(targa, 1, 16384, file1);
   fclose(file1);

   /*calculate the target coordinates based on the values in the     */
   /*upper correlation of the calibration data                       */
```

```
printf("calculating target location\n");
for(i=0; i<toptest; i++)
{
   x1=(113/(cal[1][0]-cal[0][0]))*(topx[i]-cal[0][0])+7;
   x2=(113/(cal[3][0]-cal[2][0]))*(topx[i]-cal[2][0])+7;
   y1=(30/(cal[0][1]-cal[2][1]))*(cal[0][1]-topy[i])+34;
   y2=(30/(cal[1][1]-cal[3][1]))*(cal[1][1]-topy[i])+34;
   xft[i]=(((((y1+y2)/2)-34)*(x2-7)
           + ((64-((y1+y2)/2)))*(x1-7))/30)+7;
   yft[i]=(((((x1+x2)/2)-7)*(y2-34)
           + ((120-((x1+x2)/2)))*(y1-34))/113)+34;
   xt[i]=(unsigned)(xft[i]+.5);
   yt[i]=(unsigned)(yft[i]+.5);
}

/*calculate the target coordinates based on the values in the        */
/*lower correlation of the calibration data                          */
for(i=0; i<bottest; i++)
{
   x1=(113/(cal[0][2]-cal[1][2]))*(cal[0][2]-botx[i])+7;
   x2=(113/(cal[2][2]-cal[3][2]))*(cal[2][2]-botx[i])+7;
   y1=(30/(cal[2][3]-cal[0][3]))*(boty[i]-cal[0][3])+34;
   y2=(30/(cal[3][3]-cal[1][3]))*(boty[i]-cal[1][3])+34;
   xfb[i]=(((((y1+y2)/2)-34)*(x2-7)
           + (64-((y1+y2)/2))*(x1-7))/30)+7;
   yfb[i]=(((((x1+x2)/2)-7)*(y2-34)
           + (120-((x1+x2)/2))*(y1-34))/113)+34;
   xb[i]=(unsigned)(xfb[i]+.5);
   yb[i]=(unsigned)(yfb[i]+.5);
}

eliminate_redundant();
eliminate_stray();
save_segmented_version();

/*write file with Xs on weighted background at target location*/
for(k=9; k>=0; k--)
{
   ii=0;
   for(i=123-yb[k]; i<132-yb[k]; i++)
   {
      jj=0;
      for(j=xb[k]-4; j<xb[k]+5; j++)
      {
         targa[i][j]=botvalue[k];
         if(ii==jj || ii==8-jj) targa[i][j]=0x00;
         jj++;
      }
      ii++;
   }
   ii=0;
   for(i=123-yt[k]; i<132-yt[k]; i++)
   {
         jj=0;
         for(j=xt[k]-4; j<xt[k]+5; j++)
         {
            targa[i][j]=topvalue[k];
            if(ii==jj || ii==8-jj) targa[i][j]=0x00;
            jj++;
         }
         ii++;
      }
   }

   file1=fopen("\\targa\\bullseye.tga","wb");
   fwrite(targaheader, 18, 1, file1);
   fwrite(targa, 16384, 1, file1);
   fclose(file1);
   for(i=0; i<10; i++)
   {
      printf("%u\t%u\t%u\t%u\t%u\t%u\n",xt[i], yt[i], topvalue[i],
                                          xb[i], yb[i], botvalue[i]);
   }

   printf("press any key to continue");
   getch();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
eliminate_redundant()
{
   /*eliminate some redundant marks*/
   for(k=9; k>0; k--)
   {
      for(j=k-1; j>=0; j--)
      {
         if(abs(xft[k]-xft[j])<=5 && abs(yft[k]-yft[j])<=5)
         {
            if(topvalue[j]<251) topvalue[j]+=5;
            else topvalue[j]=255;
            xt[k]=0;
            yt[k]=0;
            topvalue[k]=0;
         }
         if(abs(xfb[k]-xfb[j])<=5 && abs(yfb[k]-yfb[j])<=5)
         {
            if(botvalue[j]<251) botvalue[j]+=5;
            else botvalue[j]=255;
            xb[k]=0;
            yb[k]=0;
            botvalue[k]=0;
         }
      }
   }
}
```

```
/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
eliminate_stray()
{
   for(k=9; k>0; k--)
   {
      j=9;
      while((abs(xft[k]-xfb[j])>5 || abs(yft[k]-yfb[j])>5) && j>=0)
      {
         j--;
         if(j<0)
         {
            xt[k]=0;
            yt[k]=0;
            topvalue[k]=0;
         }
      }
      j=9;
      while((abs(xfb[k]-xft[j])>5 || abs(yfb[k]-yft[j])>5) && j>=0)
      {
         j--;
         if(j<0)
         {
            xb[k]=0;
            yb[k]=0;
            botvalue[k]=0;
         }
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
save_segmented_version()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         outbuffer[j][i]=0x00;
      }
   }

   jlow=0;
   for(j=10; (j<24 && jlow==0); j++)
   {
      for(i=32; i<97; i++)
      {
         if(targa[j][i]!=0) jlow=24-j;
      }
   }
   jhigh=0;
   for(j=38; (j>24 && jhigh==0); j--)
   {
      for(i=32; i<97; i++)
   {
      if(targa[j][i]!=0) jhigh=j-24;
   }
}
ilow=0;
for(i=32; (i<64 && ilow==0); i++)
{
   for(j=24-jlow; j<25+jhigh; j++)
   {
      if(targa[j][i]!=0) ilow=64-i;
   }
}
ihigh=0;
for(i=96; (i>64 && ihigh==0); i--)
{
   for(j=24-jlow; j<25+jhigh; j++)
   {
      if(targa[j][i]!=0) ihigh=i-64;
   }
}

for(k=0; k<10; k++)
{
   if(xb[k]!=0 && yb[k]!=0)
   {
      ii=xb[k]-ilow-5;
      if(ii<0) ii=0;
      jj=(128-yb[k])-jlow-5;
      if(jj<57) jj=57;
      for(i=ii; i<xb[k]+ihigh+6 && i<128; i++)
      {
         for(j=jj; j<(128-yb[k])+jhigh+6 && j<128; j++)
         {
            outbuffer[j][i]=0xff;
         }
      }
   }
   if(xt[k]!=0 && yt[k]!=0)
   {
      ii=xt[k]-ilow-5;
      if(ii<0) ii=0;
      jj=(128-yt[k])-jlow-5;
      if(jj<57) jj=57;
      for(i=ii; i<xt[k]+ihigh+6 && i<128; i++)
      {
         for(j=jj; j<(128-yt[k])+jhigh+6 && j<128; j++)
         {
            outbuffer[j][i]=0xff;
         }
      }
   }
}
for(j=0; j<128; j++)
```

```
   {
      for(i=0; i<55; i++)
      {
         outbuffer[i][j]=targa[i][j];
      }
      for(i=57; i<128; i++)
      {
         outbuffer[i][j]=outbuffer[i][j] & targa[i][j];
      }
   }
   file1=fopen("\\targa\\segment.tga","wb");
   fwrite(targaheader,18,1,file1);
   fwrite(outbuffer,16384,1,file1);
   fclose(file1);
}
/*********************************************************************/
```

```
/*************************************************************************/
/* DIS-HF.C                                                             */
/*                                                                      */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                     */
/*         INPUT DISPLAY TO THE JOINT TRANSFORM CORRELATOR              */
/*                     by Capt John Cline                               */
/*                      August 31, 1989                                 */
/*                                                                      */
/*This program combines two targa files, a 45x128 pixel scene and a     */
/*45x128 pixel template, to form a single file "display2.tga".  The     */
/*template is written to TARGA rows 10 through 54, and the scene to     */
/*TARGA rows 57 through 101.  The remaining areas of a 128X128 array    */
/*are left as zeros.  BOTH TARGA FILES MUST BE BINARIZED VERSIONS       */
/*SUCH THAT ONLY PIXEL VALUES OF 0X00 AND 0XFF EXIST.  After storing    */
/*the file, the array is converted to SLM format and displayed on the*/
/*SLM.                                                                  */
/*************************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];
char filename[256], filename2[256];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *file1, *file2;

void main()
{
   prepare_slm();

   /*read in template*/
   printf("Enter file name and .ext of template file");
   printf(" (in \TARGA directory).\n");
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s",filename);
   file1=fopen(filename2,"rb");
   fseek(file1,18,0);
   fread(&targabuffer[10][0],1,5760,file1);
   fclose(file1);

   /*read in scene*/
   printf("Enter file name and .ext of scene file");
   printf(" (in \TARGA directory).\n");
   scanf("%s", filename);
   sprintf(filename2,"\\targa\\%s",filename);
   file1=fopen(filename2,"rb");
   fseek(file1,18,0);
   fread(&targabuffer[57][0],1,5760 ,file1);
   fclose(file1);
   /*write a targa file for later display*/
   file2=fopen("\\targa\\display2.tga","wb");
   fwrite(targhead,18,1,file2);
   fwrite(targabuffer,1,16384,file2);
   fclose(file2);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
```

```
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*********************************************************************/
```

```
/*************************************************************************
/* FT_BOTH.C
/*
/*                    AIR FORCE INSTITUTE OF TECHNOLOGY
/*  PROGRAM FOR BINARIZING FRINGES IN THE JOINT TRANSFORM CORRELATOR
/*                         by Capt John Cline
/*                          August 31, 1989
/*
/*This program binarizes the fringes of the JTC by finding the mean
/*value of the joint transform file "grabl.tga", and using it as a
/*threshold.  If the value is greater than the threshold the pixel
/*would be on, if less, the pixel is off.  The TARGA file
/*"binary.tga" is saved with the binary version of the fringes, and
/*the SLM is appropriately written to for the next step of the JTC.
*************************************************************************

#include "stdio.h"
#include "string.h"
#include "math.h"

unsigned char targhead[18], slm[128][64];
unsigned char targabuffer[128][128], temp[128][128];
int maxvalue, i, j, x, y, col, row, index;
float combined, thrshld=0;

FILE *filel;

void main()
{
   prepare_slm();

   /*get joint transform file*/
   filel=fopen("\\targa\\grabl.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                             /*read header*/
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*calculate threshold by finding average value*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         thrshld+=targabuffer[row][col];
      }
   }
   thrshld=thrshld/16384;

   /*binarize pixels based on threshold*/
   printf("binarizing data \n");
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
```

```
        {
          if(thrshld<targabuffer[row][col]) targabuffer[row][col]=0xff;
          else targabuffer[row][col]=0x00;
        }
    }

    /*save binarized version */
    file1=fopen("\\targa\\binary.tga","wb");
    fwrite(targhead, 18, 1, file1);
    fwrite(targabuffer, 16384, 1, file1);
    fclose(file1);

    convert_targabuffer_to_temp();
    convert_temp_to_slm();
    write_to_slm();
}

/*********************************************************/
prepare_slm()
{
    char far *semetex;
    semetex = (char far *)0xb0000000;
    *(semetex+0x800)=1;                              /*clear slm*/
}

/*********************************************************/
convert_targabuffer_to_temp()
{
    for(i=0; i<128; i++)
    {
        for(j=0; j<128; j++)
        {
            if(targabuffer[i][j]==0xff) temp[i][j]=0x01;
            else temp[i][j]=0x00;
        }
    }
}

/*********************************************************/
convert_temp_to_slm()
{
    /*convert 8 pixels into an eight bit character for address in the SLM*/

    printf("creating SLM format\n");
    for(x=0; x<128; x++)
    {
        for(j=0; j<16; j++)
        {
            slm[x][4*j] = 2*temp[126-8*j][x]+temp[127-8*j][x];
            slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x];
            slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x];
            slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x];
        }
    }
}

/** ** ** ** ** ** ** ** * * * ** ** ** * * * *
write_to_slm()
{
    /*Write the SLM array to the SLM*/
    char far *semetex;
    semetex = (char far *) 0xb0000000;
    printf("writing to the SLM \n");
    for(j=0; j<16; j++)
    {
        for(x=0; x<128; x++)
        {
            i=x+128*j;
            *(semetex+i) = slm[x][4*j];
            *(semetex+i) = slm[x][1+4*j];
            *(semetex+i) = slm[x][2+4*j];
            *(semetex+i) = slm[x][3+4*j];
        }
    }
    *(semetex+0x801) = 1;
}
*****************************************************************
```

```
/*********************************************************************/
/* JT2MED.C                                                        */
/*                                                                 */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY              */
/*  PROGRAM FOR BINARIZING FRINGES IN THE JOINT TRANSFORM CORRELATOR */
/*                       by Capt John Cline                        */
/*                        August 31, 1989                          */
/*                                                                 */
/*This program binarizes the fringes of the JTC by finding the MEDIAN*/
/*value of the joint transform file "grabl.tga", and using it as a  */
/*threshold.  If the value is greater than the threshold the pixel  */
/*would be on, if less, the pixel is off.  The TARGA file           */
/*"binary.tga" is saved with the binary version of the fringes, and */
/*the SLM is appropriately written to for the next step of the JTC.  */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

unsigned char targhead[18], slm[128][64];
unsigned char targabuffer[128][128], temp[128][128];
int maxvalue, i, j, x, y, col, row, index;
int bins[256], total=0;
float combined, thrshld=0;

FILE *filel;

void main()
{
   prepare_slm();

   /*get joint transform file*/
   filel=fopen("\\targa\\grabl.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                          /*read header*/
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*calculate median of pixel values to use as a threshold*/
   printf("calculating threshold \n");
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         bins[targabuffer[row][col]]++;
      }
   }
   for(i=0; total<=8192; i++)
   {
      total+=bins[i];
      thrshld++;
   }
```

```
    /*binarize pixels based on threshold*/
    printf("binarizing data \n");
    for(row=0; row<128; row++)
    {
       for(col=0; col<128; col++)
       {
          if(thrshld<targabuffer[row][col]) targabuffer[row][col]=0xff;
          else targabuffer[row][col]=0x00;
       }
    }

    /*save binarized version */
    filel=fopen("\\targa\\binary.tga","wb");
    fwrite(targhead, 18, 1, filel);
    fwrite(targabuffer, 16384, 1, filel);
    fclose(filel);

    convert_targabuffer_to_temp();
    convert_temp_to_slm();
    write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
    char far *semetex;
    semetex = (char far *) 0xb0000000;
    *(semetex+0x800)=1;                                    /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
    for(i=0; i<128; i++)
    {
       for(j=0; j<128; j++)
       {
          if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
          else temp[i][j]=0x00;
       }
    }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_temp_to_slm()
{
    /*convert 2 pixels into an eight bit character for addressing the SLM*/

    printf("creating SLM format  \n");
    for(x=0; x<128; x++)
    {
       for(j=0; j<16; j++)
```

```
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*****************************************************************/
```

```
/*************************************************************************/
/* JT_COMB.C                                                             */
/*                                                                       */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                      */
/*  PROGRAM FOR BINARIZING FRINGES IN THE JOINT TRANSFORM CORRELATOR     */
/*                      by Capt John Cline                               */
/*                       August 31, 1989                                 */
/*                                                                       */
/*This program binarizes the fringes of the JTC by based on the sign     */
/*of difference between the joint transform file "grabl.tga" and the    */
/*Fourier transform of the scene file "top.tga", and using it as a      */
/*threshold.  If the difference is greater than the zero, the           */
/*pixel would be on, if less, the pixel is off.  The TARGA file         */
/*"binary.tga" is saved with the binary version of the fringes, and     */
/*the SLM is appropriately written to for the next step of the JTC.     */
/*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

unsigned char targhead[18], slm[128][64];
unsigned char joint[128][128], targabuffer[128][128], temp[128][128];
int maxvalue, i, j, x, y, col, row, index;
float combined;

FILE *filel;

void main()
{
   prepare_slm();

   /*get joint transform file*/
   filel=fopen("\\targa\\grabl.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                              /*read header*/
   fread(joint,1,16384,filel);
   fclose(filel);

   /*get scene only file*/
   filel=fopen("\\targa\\top.tga","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*binarize pixels based on threshold for difference*/
   printf("binarizing data \n");
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         combined=joint[row][col]-targabuffer[row][col];
         if(combined<=0)
```

```
            {
                targabuffer[row][col]=0x00;
            }
            else
            {
                targabuffer[row][col]=0xff;
            }
        }
    }

    /*save binarized version */
    filel=fopen("\\targa\\binary.tga","wb");
    fwrite(targhead, 18, 1, filel);
    fwrite(targabuffer, 16384, 1, filel);
    fclose(filel);

    convert_targabuffer_to_temp();
    convert_temp_to_slm();
    write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
    char far *semetex;
    semetex = (char far *) 0xb0000000;
    *(semetex+0x800)=1;                                      /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
    for(i=0; i<128; i++)
    {
        for(j=0; j<128; j++)
        {
            if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
            else temp[i][j]=0x00;
        }
    }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
    /*convert 2 pixels into an eight bit character for addressing the SLM*/

    printf("creating SLM format  \n");
    for(x=0; x<128; x++)
    {
        for(j=0; j<16; j++)
        {
            slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
```

```
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/************************************************************************/
```

```
/*************************************************************************/
/* DIS-HF2.C                                                             */
/*                                                                       */
/*               AIR FORCE INSTITUTE OF TECHNOLOGY                       */
/*          INPUT DISPLAY TO THE JOINT TRANSFORM CORRELATOR              */
/*                     by Capt John Cline                                */
/*                      August 31, 1989                                  */
/*                                                                       */
/*This program displays the segmented scene and template created by     */
/*an earlier run of the joint transform correlator.  The TARGA file     */
/*"segment.tga" will be read in and displayed on the SLM.               */
/*************************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *file1;

void main()
{
   prepare_slm();

   /*read in file*/
   file1=fopen("\\targa\\segment.tga","rb");
   fseek(file1,18,0);
   fread(targabuffer,1,16384,file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                  /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
            if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
            else temp[i][j]=0x00;
         }
      }
   }

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]   =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xD0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/**********************************************************************/
```

```
/***********************************************************************/
/* DISPTOP.C                                                          */
/*                                                                    */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*        SCENE ONLY DISPLAY TO THE JOINT TRANSFORM CORRELATOR        */
/*                        by Capt John Cline                          */
/*                          August 31, 1989                           */
/*                                                                    */
/*This program displays only the scene half (upper 71x128 pixels) of */
/*the TARGA file "display2.tga" onto the SLM.  This program is part  */
/*of the joint transform correlator, and is run after the scene and  */
/*template have been displayed together.                              */
/***********************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];

FILE *file1;

void main()
{
   prepare_slm();

   /*read in scene half only*/
   file1=fopen("\\targa\\display2.tga","rb");
   fseek(file1,7314,0);
   fread(&targabuffer[57][0],1,9088,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                  /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/**********************************************************************/
```

```
/**************************************************************************/
/* BULL-HF2.C                                                            */
/*                                                                       */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                    */
/*  CORRELATION PLANE INTERPRETER FOR THE JOINT TRANSFORM CORRELATOR     */
/*                       by Capt John Cline                              */
/*                        August 31, 1989                                */
/*                                                                       */
/*This program finds the top 10 values in the top and bottom cross-     */
/*correlation regions of the joint transform correlator correlation     */
/*plane file "grab2.tga".  It rewrites the correlation plane file       */
/*with crosshairs at peak locations in the cross correlation regions    */
/*to the file "max.tga".  Next, it uses these max values, along with    */
/*earlier calibration values stored in "cal.dat", to calculate the      */
/*coordinates of the target in the original scene.  (This data is       */
/*displayed on the computer monitor.)  Having done this, the original*/
/*scene file ("display2.tga) is rewritten with crosshairs at the        */
/*target location as "bullseye.tga".  This program differs from         */
/*BULL-HF.C, only in that crosshairs are written to "segment.tga"       */
/*instead of "display2.tga".  That's because this is a part of a JTC */
/*that uses SEGMENTED inputs generated from a earlier pass through      */
/*the JTC.                                                               */
/**************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, toptest=0, bottest=0, xb[10], yb[10], xt[10], yt[10];
int ii, jj, ilow, ihigh, jlow, jhigh;
unsigned char targaheader[18], targa[160][128], outbuffer[128][128];
unsigned int topx[11], topy[11], topvalue[11];
unsigned int botx[11], boty[11], botvalue[11];
unsigned char caldata[4][4];
float cal[4][4], xx, yy;
float x1, x2, y1, y2, xfb[10], xft[10], yfb[10], yft[10];

FILE *file1;

void main()
{
   /*initialize values*/
   for(i=0; i<11; i++)
   {
      topx[i]=0;
      topy[i]=0;
      topvalue[i]=0;
      botx[i]-0;
      boty[i]=0;
      botvalue[i]=0;
   }

   /*get ccd file*/
```

```
file1=fopen("\\targa\\grab2.tga","rb");
fread(targaheader,1,18,file1);
printf("\n\nreading in data \n");
fread(targa,1,20480,file1);
fclose(file1);

/*get cal data file*/
file1=fopen("\\turc\\cal.dat","rb");
fread(caldata,1,16,file1);
fclose(file1);
for(i=0; i<4; i++)
{
   for(j=0; j<4; j++)
   {
      cal[i][j]=(float)caldata[i][j];
   }
}

/*erase center autocorrelation section*/
for(i=45; i<115; i++)
{
   for(j=0; j<128; j++)
   {
      targa[i][j]=0x00;
   }
}

/*locate the 10 max values on the top*/
for(i=115; i<160; i++)
{
   for(j=0; j<128; j++)
   {
      if(targa[i][j]>=100)
      {
         if(toptest<10) toptest++;
         k=9;
         while(targa[i][j] >topvalue[k] && k>=0)
         {
            topvalue[k+1]=topvalue[k];
            topx[k+1]=topx[k];
            topy[k+1]=topy[k];
            topvalue[k]=targa[i][j];
            topx[k]=j;
            topy[k]=i;
            k--;
         }
      }
   }
}

/*locate 10 max values on the bottom*/
for(i=0; i<45; i++)
{
        for(j=0; j<128; j++)
        {
            if(targa[i][j]>=100)
            {
                if(bottest<10) bottest++;
                k=9;
                while(targa[i][j] >botvalue[k] && k>=0)
                {
                    botvalue[k+1]=botvalue[k];
                    botx[k+1]=botx[k];
                    boty[k+1]=boty[k];
                    botvalue[k]=targa[i][j];
                    botx[k]=j;
                    boty[k]=i;
                    k--;
                }
            }
        }
    }

    /*write file with crosshairs*/
    for(j=0; j<bottest; j++)
    {
        for(i=-3; i<4; i++)
        {
            targa[boty[j]+i][botx[j]]=0x00;
            targa[boty[j]][botx[j]+i]=0x00;
        }
    }
    for(j=0; j<toptest; j++)
    {
        for(i=-3; i<4; i++)
        {
            targa[topy[j]+i][topx[j]]=0x00;
            targa[topy[j]][topx[j]+i]=0x00;
        }
    }

    printf("writing correlation file\n");
    filel=fopen("\\targa\\max.tga","wb");
    fwrite(targaheader,18,1,filel);
    fwrite(targa,20480,1,filel);
    fclose(filel);

/*This part of the program finds the location of the target given the*
/*location of the correlation peak.  It puts cross hairs on the       *
/*original scene file.                                                *

    /*read in the segmented scene file*/
    printf("reading in original scene file\n");
    filel=fopen("\\targa\\segment.tga","rb");
    fread(targaheader, 1, 18, filel);
    fread(targa, 1, 16384, filel);
```

```
fclose(file1);

/*calculate the target coordinates based on the values in the
/*upper correlation of the calibration data
printf("calculating target location\n");
for(i=0; i<toptest; i++)
{
   x1=(113/(cal[1][0]-cal[0][0]))*(topx[i]-cal[0][0])+7;
   x2=(113/(cal[3][0]-cal[2][0]))*(topx[i]-cal[2][0])+7;
   y1=(3C/(cal[0][1]-cal[2][1]))*(cal[0][1]-topy[i])+34;
   y2=(30/(cal[1][1]-cal[3][1]))*(cal[1][1]-topy[i])+34;
   xft[i]=(((((y1+y2)/2)-34)*(x2-7)
           + ((64-((y1+y2)/2)))*(x1-7))/30)+7;
   yft[i]=(((((x1+x2)/2)-7)*(y2-34)
           + ((120-((x1+x2)/2)))*(y1-34))/113)+34;
   xt[i]=(unsigned)(xft[i]+.5);
   yt[i]=(unsigned)(yft[i]+.5);
}

/*calculate the target coordinates based on the values in the
/*lower correlation of the calibration data
for(i=0; i<bottest; i++)
{
   x1=(113/(cal[0][2]-cal[1][2]))*(cal[0][2]-botx[i])+7;
   x2=(113/(cal[2][2]-cal[3][2]))*(cal[2][2]-botx[i])+7;
   y1=(30/(cal[2][3]-cal[0][3]))*(boty[i]-cal[0][3])+34;
   y2=(30/(cal[3][3]-cal[1][3]))*(boty[i]-cal[1][3])+34;
   xfb[i]=(((((y1+y2)/2)-34)*(x2-7)
           + (64-((y1+y2)/2))*(x1-7))/30)+7;
   yfb[i]=(((((x1+x2)/2)-7)*(y2-34)
           + (120-((x1+x2)/2))*(y1-34))/113)+34;
   xb[i]=(unsigned)(xfb[i]+.5);
   yb[i]=(unsigned)(yfb[i]+.5);
}

eliminate_redundant();
eliminate_stray();
save_segmented_version();

/*write file with Xs on weighted background at target location
for(k=9; k>=0; k--)
{
   ii=0;
   for(i=123-yb[k]; i<132-yb[k]; i++)
   {
      jj=0;
      for(j=xb[k]-4; j<xb[k]+5; j++)
      {
         targa[i][j]=botvalue[k];
         if(ii==jj || ii==8-jj) targa[i][j]=0x00;
         jj++;
      }
      ii++;
      }
      ii=0;
      for(i=123-yt[k]; i<132-yt[k]; i++)
      {
         jj=0;
         for(j=xt[k]-4; j<xt[k]+5; j++)
         {
            targa[i][j]=topvalue[k];
            if(ii==jj || ii==8-jj) targa[i][j]=0x00;
            jj++;
         }
         ii++;
      }
   }

   file1=fopen("\\targa\\bullseye.tga","wb");
   fwrite(targaheader, 18, 1, file1);
   fwrite(targa, 16384, 1, file1);
   fclose(file1);
   for(i=0; i<10; i++)
   {
      printf("%u\t%u\t%u\t%u\t%u\t%u\n",xt[i], yt[i], topvalue[i],
                                       xb[i], yb[i], botvalue[i]);
   }

   printf("press any key to continue");
   getch();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
eliminate_redundant()
{
   /*eliminate some redundant marks*/
   for(k=9; k>0; k--)
   {
      for(j=k-1; j>=0; j--)
      {
         if(abs(xft[k]-xft[j])<=5 && abs(yft[k]-yft[j])<=5)
         {
            if(topvalue[j]<251) topvalue[j]+=5;
            else topvalue[j]=255;
            xt[k]=0;
            yt[k]=0;
            topvalue[k]=0;
         }
         if(abs(xfb[k]-xfb[j])<=5 && abs(yfb[k]-yfb[j])<=5)
         {
            if(botvalue[j]<251) botvalue[j]+=5;
            else botvalue[j]=255;
            xb[k]=0;
            yb[k]=0;
            botvalue[k]=0;
         }
```

```
            }
        }
    }

    /** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
    eliminate_stray()
    {
        for(k=9; k>0; k--)
        {
            j=9;
            while((abs(xft[k]-xfb[j])>5 || abs(yft[k]-yfb[j])>5) && j>=0)
            {
                j--;
                if(j<0)
                {
                    xt[k]=0;
                    yt[k]=0;
                    topvalue[k]=0;
                }
            }
            j=9;
            while((abs(xfb[k]-xft[j])>5 || abs(yfb[k]-yft[j])>5) && j>=0)
            {
                j--;
                if(j<0)
                {
                    xb[k]=0;
                    yb[k]=0;
                    botvalue[k]=0;
                }
            }
        }
    }

    /** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
    save_segmented_version()
    {
        for(i=0; i<128; i++)
        {
            for(j=0; j<128; j++)
            {
                outbuffer[j][i]=0x00;
            }
        }

        jlow=0;
        for(j=10; (j<24 && jlow==0); j++)
        {
            for(i=32; i<97; i++)
            {
                if(targa[j][i]!=0) jlow=24-j;
            }
        }
```

```
jhigh=0;
for(j=38; (j>24 && jhigh==0); j--)
{
   for(i=32; i<97; i++)
   {
      if(targa[j][i]!=0) jhigh=j-24;
   }
}
ilow=0;
for(i=32; (i<64 && ilow==0); i++)
{
   for(j=24-jlow; j<25+jhigh; j++)
   {
      if(targa[j][i]!=0) ilow=64-i;
   }
}
ihigh=0;
for(i=96; (i>64 && ihigh==0); i--)
{
   for(j=24-jlow; j<25+jhigh; j++)
   {
      if(targa[j][i]!=0) ihigh=i-64;
   }
}

for(k=0; k<10; k++)
{
   if(xb[k]!=0 && yb[k]!=0)
   {
      ii=xb[k]-ilow-5;
      if(ii<0) ii=0;
      jj=(128-yb[k])-jlow-5;
      if(jj<57) jj=57;
      for(i=ii; i<xb[k]+ihigh+6 && i<128; i++)
      {
         for(j=jj; j<(128-yb[k])+jhigh+6 && j<128; j++)
         {
            outbuffer[j][i]=0xff;
         }
      }
   }
   if(xt[k]!=0 && yt[k]!=0)
   {
      ii=xt[k]-ilow-5;
      if(ii<0) ii=0;
      jj=(128-yt[k])-jlow-5;
      if(jj<57) jj=57;
      for(i=ii; i<xt[k]+ihigh+6 && i<128; i++)
      {
         for(j=jj; j<(128-yt[k])+jhigh+6 && j<128; j++)
         {
            outbuffer[j][i]=0xff;
         }
         }
      }
   }
   for(j=0; j<128; j++)
   {
      for(i=0; i<55; i++)
      {
         outbuffer[i][j]=targa[i][j];
      }
      for(i=57; i<128; i++)
      {
         outbuffer[i][j]=outbuffer[i][j] & targa[i][j];
      }
   }
   file1=fopen("\\targa\\segment.tga","wb");
   fwrite(targaheader,18,1,file1);
   fwrite(outbuffer,16384,1,file1);
   fclose(file1);
}
/*********************************************************************/
```

## G.3 LPCT Software

A number of programs were written for correlation in the LPCT-$|FT|^2$ feature space. Two batch files, LRT-X.BAT and LRT-SQ.BAT, would perform all steps (except throwing the manual switch) necessary to do correlation in this feature space. These batches used an "X" and a "+", or squares of different sizes, respectively to perform the correlation.

Two additional batches, LRT-180.BAT and LRT-NEG.BAT, would take the intermediate results of the earlier two correlations (the LPCT-$|FT|^2$ features), and modify the template before performing correlation.

Of course, the batches would call on .CMD files, used with TESTTARG.EXE, and Turbo-C source code to complete their tasks. The batch files are listed below, followed by the .CMD files (in order of appearance), and the Turbo-C source code.

A number of files used in the batches are the same as those listed in sections G.1 and G.2. They will not be duplicated in this section.

```
:LRT-X.BAT
:This batch file performs a joint transform correlation on the
:log polar coordinate transform of the intensity of the Fourier
:transform of an "X" and a "+".

echo off

:CREATE THE X AND + AND DISPLAY THE X
CD \TURC
x-lrt
disptop

:CAPTURE THE FOURIER TRANSFORM OF THE X (128X128)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <ft_scene.cmd

:DISPLAY THE +
cls
CD \TURC
dispbot

:CAPTURE THE FOURIER TRANSFORM OF THE +
CD \TARGA
testtarg <ft_temp.cmd
cls

:DISPLAY A BLANK MOSLM
CD \TURC
erase-it

:CAPTURE THE FOURIER TRANSFORM OF THE BLANK TO USE AS A BIAS
CD \TARGA
testtarg <ft_erase.cmd
cls

echo THROW THE SWITCH
pause
cls

:CAPTURE THE LPCT OF THE BLANK
testtarg <110x456.cmd

:REDUCE THE LPCT OF THE BLANK TO 55X114 THEN BINARIZE THE
:FOURIER TRANSFORM OF THE X AND DISPLAY ON THE MOSLM
CD \TURC
red_lrt3
ft_scene

:CAPTURE THE LPCT OF THE X
CD \TARGA
testtarg <110x456.cmd
```

```
:REDUCE THE LPCT OF THE X TO 55X114 THEN BINARIZE THE
:FOURIER TRANSFORM OF THE + AND DISPLAY ON THE MOSLM
CD \TURC
red_lrt1
ft_temp

:CAPTURE THE LPCT OF THE +
CD \TARGA
testtarg <110x456.cmd
cls

:REDUCE THE LPCT OF THE + TO 55X114
:THEN CREATE A BINARY INPUT FRAME FOR THE JTC
CD \TURC
red_lrt2
lrt-comb

cls
echo THROW THE SWITCH
pause

:THE NEXT FEW STEPS PERFORM THE JTC

:CAPTURE THE JOINT TRANSFORM 128X128 ONLY
CD \TARGA
testtarg <jt1-lrt.cmd
cls

:DISPLAY THE TOP (THE X) ONLY
CD \TURC
top-lrt

:CAPTURE THE FOURIER TRANSFORM FOR THE X ONLY
CD \TARGA
testtarg <top-lrt.cmd
cls

:BINARIZE ON AVERAGE DIFFERENCE AND DISPLAY ON MOSLM
CD \TURC
jt-diff

:CAPTURE CORRELATION PLANE
CD \TARGA
testtarg <jt2-lrt.cmd
cls

:LOCATE PEAKS IN CORRELATION PLANE
CD \TURC
bull-lrt
CD \TARGA
testtarg <jt3-lrt.cmd
```

```
:LRT-SQ.BAT
:This batch file performs a joint transform correlation on the
:log polar coordinate transform of the intensity of the Fourier
:transform of a scene and template square.

echo off

:CREATE THE TWO SQUARES AND DISPLAY THE SCENE (TOP)
CD \TURC
disp_lrt
disptop

:CAPTURE THE FOURIER TRANSFORM OF THE SCENE (128X128)
CD \TARGA
SET TARGA=247
SET TARGASET=T8X-10G
testtarg <ft_scene.cmd

:DISPLAY THE TEMPLATE
cls
CD \TURC
dispbot

:CAPTURE THE FOURIER TRANSFORM OF THE TEMPLATE
CD \TARGA
testtarg <ft_temp.cmd
cls

:DISPLAY A BLANK MOSLM
CD \TURC
erase-it

:CAPTURE THE FOURIER TRANSFORM OF THE BLANK TO USE AS A BIAS
CD \TARGA
testtarg <ft_erase.cmd
cls

echo THROW THE SWITCH
pause
cls

:CAPTURE THE LPCT OF THE BLANK
testtarg <110x456.cmd

:REDUCE THE LPCT OF THE BLANK TO 55X114 THEN BINARIZE THE
:FOURIER TRANSFORM OF THE SCENE AND DISPLAY ON THE MOSLM
CD \TURC
red_lrt3
ft_scene

:CAPTURE THE LPCT OF THE SCENE
CD \TARGA
testtarg <110x456.cmd
```

```
:REDUCE THE LPCT OF THE SCENE TO 55X114 THEN BINARIZE THE
:FOURIER TRANSFORM OF THE TEMPLATE AND DISPLAY ON THE MOSLM
CD \TURC
red_lrt1
ft_temp

:CAPTURE THE LPCT OF THE TEMPLATE
CD \TARGA
testtarg <110x456.cmd
cls

:REDUCE THE LPCT OF THE TEMPLATE TO 55X114
:THEN CREATE A BINARY INPUT FRAME FOR THE JTC
CD \TURC
red_lrt2
lrt-comb

cls
echo THROW THE SWITCH
pause

:THE NEXT FEW STEPS PERFORM THE JTC

:CAPTURE THE JOINT TRANSFORM 128X128 ONLY
CD \TARGA
testtarg <jt1-lrt.cmd
cls

:DISPLAY THE TOP (THE SCENE) ONLY
CD \TURC
top-lrt

:CAPTURE THE FOURIER TRANSFORM FOR THE SCENE ONLY
CD \TARGA
testtarg <top-lrt.cmd
cls

:BINARIZE ON AVERAGE DIFFERENCE AND DISPLAY ON MOSLM
CD \TURC
jt-diff

:CAPTURE CORRELATION PLANE
CD \TARGA
testtarg <jt2-lrt.cmd
cls

:LOCATE PEAKS IN CORRELATION PLANE
CD \TURC
bull-lrt
CD \TARGA
testtarg <jt3-lrt.cmd
```

```
:LRT-180.BAT
:This batch file performs a joint transform correlation on the
:log polar coordinate transform of the intensity of the Fourier
:transform using feature files already created from an earlier
:run of either LRT-X.BAT or LRT-SQ.BAT.  It uses only the
:center 180 degrees of the template in the first step.

echo off
cls

:CREATE A BINARY INPUT FRAME FOR THE JTC
CD \TURC
lrt-180

:CAPTURE THE JOINT TRANSFORM 128X128 ONLY
CD \TARGA
testtarg <jtl-lrt.cmd
cls

:DISPLAY THE TOP (THE SCENE) ONLY
CD \TURC
top-lrt

:CAPTURE THE FOURIER TRANSFORM FOR THE SCENE ONLY
CD \TARGA
testtarg <top-lrt.cmd
cls

:BINARIZE ON AVERAGE DIFFERENCE AND DISPLAY ON MOSLM
CD \TURC
jt-diff

:CAPTURE CORRELATION PLANE
CD \TARGA
testtarg <jt2-lrt.cmd
cls

:LOCATE PEAKS IN CORRELATION PLANE
CD \TURC
bull-lrt
CD \TARGA
testtarg <jt3-lrt.cmd
```

```
:LRT-NEG.BAT
:This batch file performs a joint transform correlation on the
:log polar coordinate transform of the intensity of the Fourier
:transform using feature files already created from an earlier
:run of either LRT-X.BAT or LRT-SQ.BAT.  It uses only the
:center 180 degrees of the NEGATIVE of the template in the
:first step.

echo off
cls

:CREATE A BINARY INPUT FRAME FOR THE JTC
CD \TURC
lrt-neg

:CAPTURE THE JOINT TRANSFORM 128X128 ONLY
CD \TARGA
testtarg <jt1-lrt.cmd
cls

:DISPLAY THE TOP (THE SCENE) ONLY
CD \TURC
top-lrt

:CAPTURE THE FOURIER TRANSFORM FOR THE SCENE ONLY
CD \TARGA
testtarg <top-lrt.cmd
cls

:BINARIZE ON AVERAGE DIFFERENCE AND DISPLAY ON MOSLM
CD \TURC
jt-diff

:CAPTURE CORRELATION PLANE
CD \TARGA
testtarg <jt2-lrt.cmd
cls

:LOCATE PEAKS IN CORRELATION PLANE
CD \TURC
bull-lrt
CD \TARGA
testtarg <jt3-lrt.cmd
```

```
:FT_SCENE.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the JTC using LPCT features to capture a 128X128
:pixel region.  The data is stored in the file "ft_scene.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 128X128
putpic
ft_scene
192
136
319
263
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
ft_scene
0
0
-1
graphend
quit
```

```
:FT_TEMP.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the JTC using LPCT features to capture a 128X128
:pixel region.  The data is stored in the file "ft_temp.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 128X128
putpic
ft_temp
192
136
319
263
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
ft_temp
0
0
-1
graphend
quit
```

```
:FT_ERASE.CMD
:This is a command file to be used with the TARGA program TESTTAP.EXE
:It is used in the JTC using LPCT features to capture a 128X128
:pixel region.  The data is stored in the file "ft_erase.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME [illegible]
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 128X128
putpic
ft_erase
192
136
319
263
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
ft_erase
0
0
-1
graphend
quit
```

```
:JT1-LRT.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 128X128
:pixel region.  The data is stored in the file "grabl.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 128x128
putpic
grabl
192
136
319
263
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
grabl
0
0
-1
graphend
quit
```

```
:TOP-LRT.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 128X128
:pixel region.  The data is stored in the file "top.tga".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 128x128
putpic
top
192
136
319
263
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
top
0
0
-1
graphend
quit
```

```
:JT2-LRT.CMD
:This is a command file to be used with the TARGA program TESTTARG.EXE.
:It is used in the joint transform correlator to capture a 200X128
:pixel region.  The data is stored in the file "corr.lrt".

live
:ERASURES PROVIDE A DELAY AFTER GOING LIVE TO ENSURE PROPER FRAME GRAB
erase
0
0
0
erase
0
0
0

:GRAB THE FRAME
grab
dis

:SAVE CENTER 200X128
putpic
corr.lrt
192
100
319
299
-1
erase
0
0
0

:REDISPLAY SAVED SECTION
getpic
corr.lrt
0
0
-1
graphend
quit
```

```
:JT3-LRT.CMD
:This is a command file to be used with tne TARGA program TESTTARG.EXE.
:It is used to display the results of the joint transform correlator
:using the LPCT features.

erase
0
0
0
dis

:GET OPTICAL FOURIER OF SCENE
getpic
ft_scene
0
271
-1

:GET BINARY FOURIER OF SCENE
getpic
ft_scene.bin
128
271
-1

:GET OPTICAL FOURIER OF TEMPLATE
getpic
ft_temp
256
271
-1

:GET BINARY FOURIER OF TEMPLATE
getpic
ft_temp.bin
384
271
-1

:GET ORIGINAL INPUT FILE
getpic
display2
0
135
-1

:GET OPTICAL LPCT OF FT OF SCENE
getpic
scene.lrt
128
195
-1
```

```
:GET OPTICAL LPCT OF FT OF TEMPLATE
getpic
template.lrt
128
135
-1

:GET BINARY LPCTs IN JTC INPUT FORMAT
getpic
lrt_comb
256
135
-1

:GET JOINT TRANSFORM OF TWO LPCT INPUTS
getpic
grabl
0
0
-1

:GET FT OF SCENE LPCT ALONE
getpic
top
128
0
-1

:BINARY JOINT TRANSFORM
getpic
binary
256
0
-1

:GET CORRELATION PLANE
getpic
corr.lrt
384
0
-1

:SAVE RESULTS
putpic
lrt_disp
0
0
511
399
-1
graphend
quit
```

```
/**********************************************************************/
/* X-LRT.C                                                            */
/*                                                                    */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*         INPUT FILE MAKER FOR THE COORDINATE TRANSFORM JTC          */
/*                      by Capt John Cline                            */
/*                      September 30, 1989                            */
/*                                                                    */
/*This program creates an array with a plus and an X, and saves them */
/*in the TARGA file "display2.tga".  This file will be used to       */
/*correlate in the coordinate transform feature space.  The programs */
/*DISPTOP.C (used in the JTC) and DISPBOT.C will display these two   */
/*patterns on the MOSLM one at a time, so their Fourier features can */
/*be collected.                                                       */
/**********************************************************************/

#include "stdio.h"
#include "string.h"

int col, row;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};
unsigned char  targa[128][128];

FILE *file1;

void main()
{
   /*write the scene "x"*/
   for(col=47; col<81; col++)
   {
      targa[col+40][col-1]=255;
      targa[167-col][col-1]=255;
      targa[col+40][col]=255;
      targa[167-col][col]=255;
      targa[col+40][col+1]=255;
      targa[167-col][col+1]=255;
   }

   /*write the template "+"*/
   for(col=40; col<88; col++)
   {
      targa[31][col]=255;
      targa[32][col]=255;
   }
   for(row=7; row<55; row++)
   {
      targa[row][63]=255;
      targa[row][64]=255;
   }
   /*save TARGA file*/
   printf("writing TARGA file to targa\\display2.tga\n");
   filel=fopen("\\targa\\display2.tga","wb");
   fwrite(targhead, 18, 1, filel);
   fwrite(targa, 16384, 1, filel);
   fclose(filel);
}
/**********************************************************************/
```

```
/*********************************************************************/
/* DISPBOT.C                                                         */
/*                                                                   */
/*                 AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*     TEMPLATE ONLY DISPLAY TO THE JOINT TRANSFORM CORRELATOR       */
/*                      by Capt John Cline                           */
/*                      September 30, 1989                           */
/*                                                                   */
/*This program displays only the template half (lower 57x128 pixels) */
/*of the TARGA file "display2.tga" onto the upper portion of the SLM.*/
/*This program is part of the coordinate transform joint transform   */
/*correlator, and is used to capture the Fourier features of the     */
/*template.                                                          */
/*********************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];

FILE *file1;

void main()
{
   prepare_slm();

   /*read in template half only*/
   file1=fopen("\\targa\\display2.tga","rb");
   fseek(file1,18,0);
   fread(&targabuffer[83][0],1,5760,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                  /*clear slm*
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = ( char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/**********************************************************************/
```

```
/********************************************************************/
/* ERASE-IT.C                                                       */
/*                                                                  */
/*               AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/*                      MOSLM ERASE PROGRAM                         */
/*                      by Capt John Cline                          */
/*                        August 31, 1989                           */
/*                                                                  */
/*This program erases the MOSLM, that's all.                        */
/********************************************************************/

#include "stdio.h"
#include "string.h"

void main()
{
   /*erase the slm*/
   char far *semetex;
   printf("erasing the SLM\n");
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                  /*clear slm*/
}
/********************************************************************/
```

```
/************************************************************************/
/* RED_LRT3.C                                                           */
/*                                                                      */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*  PROGRAM FOR REDUCING FILES USED IN THE COORDINATE TRANSFORM JTC     */
/*                        by Capt John Cline                            */
/*                        September 30, 1989                            */
/*                                                                      */
/*This program reduces a 110x456 TARGA file to 55x128 for display on    */
/*the SLM in the joint transform correlator. The reduction is done      */
/*by averaging 2 for 1 vertically and 4 for 1 horizontally.  Zeros      */
/*are added on the left and right sides.  The file is stored as         */
/*"erase.lrt". This new file is a reduced version of the log polar      */
/*coordinate transform of an erased MOSLM.  It will be used to          */
/*reduce system bias during binarization of the scene and template      */
/*LPCT Fourier features.                                                */
/************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x37', '\x00', '\x08', '\x00'};
unsigned char inbuffer[110][456], outbuffer[55][128];
int combined, i, j, k, ii, jj;

FILE *file1, *file2;

void main()
{
  printf("reducing lrt of erased SLM\n");

  /*open files and handle headers*/
  file1=fopen("\\targa\\grablrt.tga","rb");
  fseek(file1,18,0);
  file2=fopen("\\targa\\erase.lrt","wb");
  fwrite(targhead, 18, 1, file2);

  /*read data in, reduce, and output*/
  fread(inbuffer, 1, 50160, file1);
  for(i=0; i<110; i+=2)
  {
    for(j=0; j<456; j+=4)
    {
      combined=0;
      for(ii=i; ii<i+2; ii++)
      {
        for(jj=j; jj<j+4; jj++)
        {
          combined+=inbuffer[ii][jj];
        }
```

```
         }
         outbuffer[i/2][j/4+7]=(combined+4)/8;
      }
   }
   fwrite(outbuffer, 7040, 1, file2);
   fclose(file1);
   fclose(file2);
}
```

```
/************************************************************************/
/* FT_SCENE.C                                                           */
/*                                                                      */
/*                    AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*         PROGRAM FOR BINARIZING FRINGES THE FOURIER TRANSFORM         */
/*                          by Capt John Cline                          */
/*                          September 30, 1989                          */
/*                                                                      */
/*This program binarizes the Fourier transform of the original scene */
/*input to a coordinate transform joint transform correlator (before */
/*coordinate transforming).  There are two input files "ft_scene.tga"*/
/*and the system bias file "ft_erase.tga" (created by detecting the  */
/*Fourier transform of a blank MOSLM).  The bias file values are     */
/*subtracted from those of the scene file and the result is binarized*/
/*on five times the average difference (you may want to look at      */
/*alternative binarization techniques).  The binary array is stored  */
/*in TARGA format as "ft_scene.bin" and is displayed to the MOSLM for*/
/*log polar coordinate transformation.                               */
/************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int maxvalue, i. j, k, test, x, y. col, row  index;
float combined, thrshld=0;
unsigned char bufferin[128][128], temp[128][128];
unsigned char targabuffer[128][128], slm[128][64];
char targhead[18];

FILE *filel;

void main()
{
   prepare_slm();

   /*get scene file*/
   filel=fopen("\\targa\\ft_scene.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                              /*read header*/
   fread(bufferin,1,16384,filel);
   fclose(filel);

   /*get erase file*/
   filel=fopen("\\targa\\ft_erase.tga","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*calculate threshold by finding average difference*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
```

```
        {
           thrshld+=bufferin[row][col]-targabuffer[row][col];
        }
     }
     thrshld=5*thrshld/16384;

     /*binarize pixels based on threshold for difference*/
     printf("binarizing data \n");
     for(row=0; row<128; row++)
     {
        for(col=0; col<128; col++)
        {
           combined=bufferin[row][col]-targabuffer[row][col];
           if(combined<=thrshld)
           {
              targabuffer[row][col]=0x00;
           }
           else
           {
              targabuffer[row][col]=0xff;
           }
        }
     }

     /*save binarized version */
     file1=fopen("\\targa\\ft_scene.bin","wb");
     fwrite(targhead, 18, 1, file1);
     fwrite(targabuffer, 16384, 1, file1);
     fclose(file1);

     convert_targabuffer_to_temp();
     convert_temp_to_slm();
     write_to_slm();
  }

  /**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
  prepare_slm()
  {
     char far *semetex;
     semetex = (char far *) 0xb0000000;
     *(semetex+0x800)=1;                                    /*clear slm*
  }

  /**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
  convert_targabuffer_to_temp()
  {
     for(i=0; i<128; i++)
     {
        for(j=0; j<128; j++)
        {
           if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
           else temp[i][j]=0x00;
        }
```

```
    }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
convert_temp_to_slm()
{
    /*convert 2 pixels into an eight bit character for addressing the SLM*

    printf("creating SLM format  \n");
    for(x=0; x<128; x++)
    {
        for(j=0; j<16; j++)
        {
            slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
            slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
            slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
            slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
        }
    }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
    /*Write the SLM array to the SLM*/
    char far *semetex;
    semetex = (char far *) 0xb0000000;
    printf("writing to the SLM \n");
    for(j=0; j<16; j++)
    {
        for(x=0; x<128; x++)
        {
            i=x+128*j;
            *(semetex+i) = slm[x][4*j];
            *(semetex+i) = slm[x][1+4*j];
            *(semetex+i) = slm[x][2+4*j];
            *(semetex+i) = slm[x][3+4*j];
        }
    }
    *(semetex+0x801)=1;
}
/***************************************************************************
```

```
/*********************************************************************/
/* RED_LRT1.C                                                        */
/*                                                                   */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                */
/*  PROGRAM FOR REDUCING FILES USED IN THE COORDINATE TRANSFORM JTC  */
/*                        by Capt John Cline                         */
/*                        September 30, 1989                         */
/*                                                                   */
/*This program reduces a 110x456 TARGA file to 55x128 for display on */
/*the SLM in the joint transform correlator.  The reduction is done  */
/*by averaging 2 for 1 vertically and 4 for 1 horizontally.  Zeros   */
/*are added on the left and right sides.  The file is stored as      */
/*"scene.lrt".  This new file is a reduced version of the log polar  */
/*coordinate transform of a scene's Fourier features.  It will be    */
/*binarized in another program, then used as an input to the joint   */
/*transform correlator.                                              */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x37', '\x00', '\x08', '\x00'};
unsigned char inbuffer[110][456], outbuffer[55][128];
int combined, i, j, k, ii, jj;

FILE *file1, *file2;

void main()
{
   printf("reducing lrt of scene\n");

   /*open files and handle headers*/
   file1=fopen("\\targa\\grablrt.tga","rb");
   fseek(file1,18,0);
   file2=fopen("\\targa\\scene.lrt","wb");
   fwrite(targhead, 18, 1, file2);

   /*read data in, reduce, and output*/
   fread(inbuffer, 1, 50160, file1);
   for(i=0; i<110; i+=2)
   {
      for(j=0; j<456; j+=4)
      {
         combined=0;
         for(ii=i; ii<i+2; ii++)
         {
            for(jj=j; jj<j+4; jj++)
            {
               combined+=inbuffer[ii][jj];
            }
         }
         outbuffer[i/2][j/4+7]=(combined+4)/8;
      }
   }
   fwrite(outbuffer, 7040, 1, file2);
   fclose(file1);
   fclose(file2);
}
```

```
/**********************************************************************/
/* FT_TEMP.C                                                          */
/*                                                                    */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*      PROGRAM FOR BINARIZING FRINGES THE FOURIER TRANSFORM          */
/*                        by Capt John Cline                          */
/*                        September 30, 1989                          */
/*                                                                    */
/*This program binarizes the Fourier transform of the original scene */
/*input to a coordinate transform joint transform correlator (before */
/*coordinate transforming).  There are two input files "ft_temp.tga" */
/*and the system bias file "ft_erase.tga" (created by detecting the  */
/*Fourier transform of a blank MOSLM).  The bias file values are     */
/*subtracted from those of the scene file and the result is binarized*/
/*on five times the average difference (you may want to look at      */
/*alternative binarization techniques).  The binary array is stored  */
/*in TARGA format as "ft_temp.bin" and is displayed to the MOSLM for */
/*log polar coordinate transformation.                               */
/**********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int maxvalue, i, j, k, test, x, y, col, row, index;
float combined, thrshld=0;
unsigned char bufferin[128][128], temp[128][128];
unsigned char targabuffer[128][128], slm[128][64];
char targhead[18];

FILE *filel;

void main()
{
   prepare_slm();

   /*get template file*/
   filel=fopen("\\targa\\ft_temp.tga","rb");
   printf("\n\nreading in data \n");
   fread(targhead,1,18,filel);                        /*read header*/
   fread(bufferin,1,16384,filel);
   fclose(filel);

   /*get erase file*/
   filel=fopen("\\targa\\ft erase.tga","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,16384,filel);
   fclose(filel);

   /*calculate threshold by finding average difference*/
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         thrshld+=bufferin[row][col]-targabuffer[row][col];
      }
   }
   thrshld=5*thrshld/16384;

   /*binarize pixels based on threshold for difference*/
   printf("binarizing data \n");
   for(row=0; row<128; row++)
   {
      for(col=0; col<128; col++)
      {
         combined=bufferin[row][col]-targabuffer[row][col];
         if(combined<=thrshld)
         {
            targabuffer[row][col]=0x00;
         }
         else
         {
            targabuffer[row][col]=0xff;
         }
      }
   }

   /*save binarized version */
   file1=fopen("\\targa\\ft_temp.bin","wb");
   fwrite(targhead, 18, 1, file1);
   fwrite(targabuffer, 16384, 1, file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                     /*clear slm*/
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
```

```
    }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*******************************************************************
```

```
/*********************************************************************/
/* RED_LRT2.C                                                        */
/*                                                                   */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                */
/*  PROGRAM FOR REDUCING FILES USED IN THE COORDINATE TRANSFORM JTC  */
/*                       by Capt John Cline                          */
/*                       September 30, 1989                          */
/*                                                                   */
/*This program reduces a 110x456 TARGA file to 55x128 for display on */
/*the SLM in the joint transform correlator. The reduction is done   */
/*by averaging 2 for 1 vertically and 4 for 1 horizontally.  Zeros   */
/*are added on the left and right sides.  The file is stored as      */
/*"template.lrt".  This new file is a reduced version of the log     */
/*polar coordinate transform of a template's Fourier features.  It   */
/*will be binarized in another program, then used as an input to the */
/*joint transform correlator.                                        */
/*********************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x37', '\x00', '\x08', '\x00'};
unsigned char inbuffer[110][456], outbuffer[55][128];
int combined, i, j, k, ii, jj;

FILE *file1, *file2;

void main()
{
   printf("reducing lrt of template\n");

   /*open files and handle headers*/
   file1=fopen("\\targa\\grablrt.tga","rb");
   fseek(file1,18,0);
   file2=fopen("\\targa\\template.lrt","wb");
   fwrite(targhead, 18, 1, file2);

   /*read data in, reduce, and output*/
   fread(inbuffer, 1, 50160, file1);
   for(i=0; i<110; i+=2)
   {
      for(j=0; j<456; j+=4)
      {
         combined=0;
         for(ii=i; ii<i+2; ii++)
         {
            for(jj=j; jj<j+4; jj++)
            {
               combined+=inbuffer[ii][jj];
            }
         }
         outbuffer[i/2][j/4+7]=(combined+4)/8;
      }
   }
   fwrite(outbuffer, 7040, 1, file2);
   fclose(file1);
   fclose(file2);
}
```

```
/**********************************************************************/
/* LRT-COMB.C                                                         */
/*                                                                    */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                   */
/*       INPUT DISPLAY TO LOG POLAR JOINT TRANSFORM CORRELATOR        */
/*                       by Capt John Cline                           */
/*                       SEPTEMBER 30, 1989                           */
/*                                                                    */
/*This program combines two 55x128 pixel TARGA files, "scene.lrt" and*/
/*"template.lrt", to form a single binary file "lrt_comb.tga".  These*/
/*two grey scale input files represent the log polar coordinate      */
/*transform of the magnitude of the Fourier transform of a scene and */
/*template respectively.  A third frame "erase.lrt" is the coordinate*/
/*transform of an erased MOSLM, and represents a 'system bias'.  This*/
/*system bias is subtracted from each of the other two inputs, and   */
/*binarization is based on the mean value of the active region       */
/*(a 55X114 pixel area in each file) of these two files after the    */
/*subtraction.  Zeros are used to fill in various unused regions of  */
/*the 128X128 "lrt_comb.tga" file.  After creating this binary file, */
/*it is written to the SLM to begin a joint transform correlation    */
/*using coordinate transformed magnitude of the Fourier transform    */
/*features.
     */
/**********************************************************************/

#include "stdio.h"
#include "string.h"

float thrshld=0;
int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], erased[55][128];
unsigned char temp[128][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *filel;

void main()
{
   prepare_slm();

   printf("reading data\n");
   filel=fopen("\\targa\\template.lrt","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,7040,filel);
   fclose(filel);

   filel=fopen("\\targa\\scene.lrt","rb");
   fseek(filel,18,0);
   fread(&targabuffer[73][0],1,7040,filel);
   fclose(filel);
```

```
file1=fopen("\\targa\\erase.lrt","rb");
fseek(file1,18,0);
fread(erased,1,7040,file1);
fclose(file1);

printf("calculating thresholds and binarizing\n");
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i][j]-erased[i][j])
         targabuffer[i][j]=0xff;
      else targabuffer[i][j]=0x00;
   }
}

thrshld=0;
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i+73][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i+73][j]-erased[i][j])
         targabuffer[i+73][j]=0xff;
      else targabuffer[i+73][j]=0x00;
   }
}

for(j=0; j<7; j++)
{
   for(i=0; i<55; i++)
   {
      targabuffer[i][j]=0x00;
      targabuffer[i][j+121]=0x00;
      targabuffer[i+73][j]=0x00;
      targabuffer[i+73][j+121]=0x00;
```

```
         }
      }

      file1=fopen("\\targa\\lrt_comb.tga","wb");
      fwrite(targhead,18,1,file1);
      fwrite(targabuffer,1,16384,file1);
      fclose(file1);

      convert_targabuffer_to_temp();
      convert_temp_to_slm();
      write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                       /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]   =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
  }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
```

```
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*********************************************************************/
```

```
/*************************************************************************/
/* TOP-LRT.C                                                             */
/*                                                                       */
/*                AIR FORCE INSTITUTE OF TECHNOLOGY                      */
/*      SCENE ONLY DISPLAY TO THE JOINT TRANSFORM CORRELATOR             */
/*                      by Capt John Cline                               */
/*                      SEPTEMBER 30, 1989                               */
/*                                                                       */
/*This program displays only the scene half (upper 71x128 pixels) of    */
/*the TARGA file "lrt_comb.tga" onto the SLM.  This program is part     */
/*of the coordinate transform joint transform correlator, and is run    */
/*after the scene and template have been displayed together.            */
/*************************************************************************/

#include "stdio.h"
#include "string.h"

int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], temp[128][128];

FILE *file1;

void main()
{
   prepare_slm();

   /*read in scene half only*/
   file1=fopen("\\targa\\lrt_comb.tga","rb");
   fseek(file1,7314,0);
   fread(&targabuffer[57][0],1,9088,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                      /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
```

```
            else temp[i][j]=0x00;
         }
      }
   }

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*/

   printf("creating SLM format  \n");
   foı(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j].
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/***************************************************************************
```

```
/*****************************************************************************/
/* BULL-LRT.C                                                              */
/*                                                                         */
/*                 AIR FORCE INSTITUTE OF TECHNOLOGY                       */
/*    CORRELATION PEAK LOCATOR FOR THE JOINT TRANSFORM CORRELATOR          */
/*             USING LOG POLAR COORDINATE TRANSFORM INPUTS                 */
/*                          by Capt John Cline                             */
/*                          September 30, 1989                             */
/*                                                                         */
/*This program finds the top 15 values in the top and bottom cross-        */
/*correlation regions of the joint transform correlator correlation        */
/*plane file "corr.lrt".  It rewrites the correlation plane file with*/
/*crosshairs at these peak locations in the cross correlation regions*/
/*to the file "max.lrt".  It also displays the coordinates and values*/
/*of these peaks on the computer monitor.                                  */
/*****************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int i, j, k, ii, jj, top est=0, bottest=0;
int xb[15], yb[15], xt[15], yt[15];
int ilow, ihigh, jlow, jhigh;
unsigned char targhead[18], targabuffer[200][128];
unsigned int topx[16], topy[16], topvalue[16];
unsigned int botx[16], boty[16], botvalue[16];

FILE *file1;

void main()
{
   /*initialize values*/
   for(i=0; i<16; i++)
   {
      topx[i]=0;
      topy[i]=0;
      topvalue[i]=0;
      botx[i]=0;
      boty[i]=0;
      botvalue[i]=0;


   /*get ccd file*/
   file1=fopen("\starga\corr.lrt","rb");
   fread(targhead,1,18,file1);
   printf("\n reading in data \n");
   fread(targabuffer,1,25600,file1);
   fclose(file1);

   /*erase center auto correlation section*/
   for(i=[illegible]; i<[illegible]; i++)
```

```
    for(j=0; j<128; j++)
    {
        targabuffer[i][j]=0x00;
    }
}

/*locate the 15 max values on the top*/
for(i=150; i<200, i++)
{
    for(j=0; j<128; j++)
    {
        if(targabuffer[i][j]>=100)
        {
            if(toptest<15) toptest++;
            k=14;
            while(targabuffer[i][j] >topvalue[k] && k>=0)
            {
                topvalue[k+1]=topvalue[k];
                topx[k+1]=topx[k];
                topy[k+1]=topy[k];
                topvalue[k]=targabuffer[i][j];
                topx[k]=j;
                topy[k]=i;
                k--;
            }
        }
    }
}

/*locate 15 max values on the bottom*/
for(i=0; i<50; i++)
{
    for(j=0; j<128; j++)
    {
        if(targabuffer[i][j]>=100)
        {
            if(bottest<15) bottest++;
            k=14;
            while(targabuffer[i][j] >botvalue[k] && k>=0)
            {
                botvalue[k+1]=botvalue[k];
                botx[k+1]=botx[k];
                boty[k+1]=boty[k];
                botvalue[k]=targabuffer[i][j];
                botx[k]=j;
                boty[k]=i;
                k--;
            }
        }
    }
}

/*write file with crosshairs*/
```

```
    for(j=0; j<bottest; j++)
    {
       for(i=-3; i<4; i++)
       {
          targabuffer[boty[j]+i][botx[j]]=0x00;
          targabuffer[boty[j]][botx[j]+i]=0x00;
       }
    }
    for(j=0; j<toptest; j++)
    {
       for(i=-3; i<4; i++)
       {
          targabuffer[topy[j]+i][topx[j]]=0x00;
          targabuffer[topy[j]][topx[j]+i]=0x00;
       }
    }

    printf("writing correlation file\n");
    file1=fopen("\\targa\\max.lrt","wb");
    fwrite(targhead,18,1,file1);
    fwrite(targabuffer,25600,1,file1);
    fclose(file1);

    for(i=0; i<15; i++)
    {
       printf("%u\t%u\t%u\t%u\t%u\t%u\n",topx[i], topy[i], topvalue[i],
                                         botx[i], boty[i], botvalue[i]);
    }

    printf("press any key to continue");
    getch();
}
/*****************************************************************
```

```
/*************************************************************************/
/* DISP_LRT.C                                                            */
/*                                                                       */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                    */
/*         INPUT FILE MAKER FOR THE COORDINATE TRANSFORM JTC             */
/*                        by Capt John Cline                             */
/*                        September 30, 1989                             */
/*                                                                       */
/*This program creates an array with two squares, one of which (the     */
/*scene) whose size the user can specify, and saves them in the TARGA*/
/*file "display2.tga".  This file will be used to correlate in the      */
/*coordinate transform feature space.  The programs DISPTOP.C (used     */
/*in the JTC) and DISPBOT.C will display these two patterns on the      */
/*MOSLM one at a time, so their Fourier features can be collected.      */
/*************************************************************************/

#include "stdio.h"
#include "string.h"
#include "math.h"

int col, row, length;
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};
unsigned char targa[128][128];

FILE *filel;

void main()
{
   /*get square lengths and open file*/
   printf("Enter scene square length of side (odd & max of 45):\n");
   scanf("%d",&length);

   printf("calculating\n");
   /*zero the matrix*/
   for(col=0; col<128; col++)
   {
      for(row=0; row<128; row++)
      {
         targa[row][col]=0;
      }
   }

   /*write the scene square*/
   for(col=64-(length-1)/2; col<65+(length-1)/2; col++)
   {
      for(row=105-(length-1)/2; row<106+(length-1)/2; row++)
      {
         targa[row][col]=255;
      }
   }
   /*write the template square*/
   for(col=57; col<72; col++)
   {
      for(row=15; row<30; row++)
      {
         targa[row][col]=255;
      }
   }

   /*save TARGA file*/
   printf("writing TARGA file to targa\\display2.tga\n");
   file1=fopen("\\targa\\display2.tga","wb");
   fwrite(targhead, 18, 1, file1);
   fwrite(targa, 16384, 1, file1);
   fclose(file1);
}
/**********************************************************************/
```

```
/*********************************************************************/
/* LRT-180.C                                                         */
/*                                                                   */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                */
/*        INPUT DISPLAY TO LOG POLAR JOINT TRANSFORM CORRELATOR      */
/*                         by Capt John Cline                        */
/*                         SEPTEMBER 30, 1989                        */
/*                                                                   */
/*This program combines two 55x128 pixel TARGA files, "scene.lrt" and*/
/*"template.lrt", to form a single binary file "lrt_comb.tga".  These*/
/*two grey scale input files represent the log polar coordinate      */
/*transform of the magnitude of the Fourier transform of a scene and */
/*template respectively.  A third frame "erase.lrt" is the coordinate*/
/*transform of an erased MOSLM, and represents a 'system bias'.  This*/
/*system bias is subtracted from each of the other two inputs, and   */
/*binarization is based on the mean value of the active region       */
/*(a 55X114 pixel area in each file) of these two files after the    */
/*subtraction.  Zeros are used to fill in various unused regions of  */
/*the 128X128 "lrt_comb.tga" file.  After creating this binary file, */
/*it is written to the SLM to begin a joint transform correlation    */
/*using coordinate transformed magnitude of the Fourier transform    */
/*features.  THE DIFFERENCE BETWEEN THIS AND LRT-COMB IS THAT ONLY   */
/*ABOUT THE CENTER 180 DEGREES OF THE 360 TEMPLATE IS USED.          */
/*********************************************************************/

#include "stdio.h"
#include "string.h"

float thrshld=0;
int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], erased[55][128];
unsigned char temp[128][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *filel;

void main()
{
   prepare_slm();

   printf("reading data\n");
   filel=fopen("\\targa\\template.lrt","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,7040,filel);
   fclose(filel);

   filel=fopen("\\targa\\scene.lrt","rb");
   fseek(filel,18,0);
   fread(&targabuffer[73][0],1,7040,filel);
   fclose(filel);
```

```
file1=fopen("\\targa\\erase.lrt","rb");
fseek(file1,18,0);
fread(erased,1,7040,file1);
fclose(file1);

printf("calculating thresholds and binarizing\n");
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<35; j++)
{
   for(i=0; i<55; i++)
   {
      targabuffer[i][j]=0x00;
      targabuffer[i][127-j]=0x00;
   }
}

for(j=35; j<93; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i][j]-erased[i][j])
         targabuffer[i][j]=0xff;
      else targabuffer[i][j]=0x00;
   }
}

thrshld=0;
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i+73][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i+73][j]-erased[i][j])
         targabuffer[i+73][j]=0xff;
      else targabuffer[i+73][j]=0x00;
   }
}
```

```
   for(j=0; j<7; j++)
   {
      for(i=0; i<55; i++)
      {
         targabuffer[i][j]=0x00;
         targabuffer[i][j+121]=0x00;
         targabuffer[i+73][j]=0x00;
         targabuffer[i+73][j+121]=0x00;
      }
   }

   /*write the binary file*/
   file1=fopen("\\targa\\lrt_comb.tga","wb");
   fwrite(targhead,18,1,file1);
   fwrite(targabuffer,1,16384,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000:
   *(semetex+0x800)=1;                                    /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
   {
      for(j=0; j<16; j++)
      {
         slm[x][4*j]  =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
         slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
         slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
         slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
      }
   }
}

/** ** ** ** ** ** ** ** * * * ** ** ** ** ** ** ** **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*******************************************************************/
```

```
/**********************************************************************/
/* LRT-NEG.C                                                          */
/*                                                                    */
/*                  AIR FORCE INSTITUTE OF TECHNOLOGY                 */
/*      INPUT DISPLAY TO LOG POLAR JOINT TRANSFORM CORRELATOR         */
/*                        by Capt John Cline                          */
/*                        SEPTEMBER 30, 1989                          */
/*                                                                    */
/*This program combines two 55x128 pixel TARGA files, "scene.lrt" and*/
/*"template.lrt", to form a single binary file "lrt_comb.tga".  These*/
/*two grey scale input files represent the log polar coordinate      */
/*transform of the magnitude of the Fourier transform of a scene and */
/*template respectively.  A third frame "erase.lrt" is the coordinate*/
/*transform of an erased MOSLM, and represents a 'system bias'.  This*/
/*system bias is subtracted from each of the other two inputs, and   */
/*binarization is based on the mean value of the active region       */
/*(a 55X114 pixel area in each file) of these two files after the    */
/*subtraction.  Zeros are used to fill in various unused regions of  */
/*the 128X128 "lrt_comb.tga" file.  After creating this binary file, */
/*it is written to the SLM to begin a joint transform correlation    */
/*using coordinate transformed magnitude of the Fourier transform    */
/*features.  THE DIFFERENCES BETWEEN THIS AND LRT-COMB ARE THAT ONLY */
/*ABOUT THE CENTER 180 DEGREES OF THE 360 TEMPLATE IS USED, AND THE  */
/*TEMPLATE IS NEGATIVE.                                              */
/**********************************************************************/

#include "stdio.h"
#include "string.h"

float thrshld=0;
int i, j, x;
unsigned char slm[128][64], targabuffer[128][128], erased[55][128];
unsigned char temp[128][128];
char targhead[] = {'\x00', '\x00', '\x03', '\x00', '\x00', '\x00',
                   '\x00', '\x00', '\x00', '\x00', '\x00', '\x00',
                   '\x80', '\x00', '\x80', '\x00', '\x08', '\x00'};

FILE *filel;

void main()
{
   prepare_slm();

   printf("reading data\n");
   filel=fopen("\\targa\\template.lrt","rb");
   fseek(filel,18,0);
   fread(targabuffer,1,7040,filel);
   fclose(filel);

   filel=fopen("\\targa\\scene.lrt","rb");
   fseek(filel,18,0);
   fread(&targabuffer[73][0],1,7040,filel);
   fclose(filel);
```

```
file1=fopen("\\targa\\erase.lrt","rb");
fseek(file1,18,0);
fread(erased,1,7040,file1);
fclose(file1);

printf("calculating thresholds and binarizing\n");
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<35; j++)
{
   for(i=0; i<55; i++)
   {
      targabuffer[i][j]=0x00;
      targabuffer[i][127-j]=0x00;
   }
}

for(j=35; j<93; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i][j]-erased[i][j])
         targabuffer[i][j]=0x00;
      else targabuffer[i][j]=0xff;
   }
}

thrshld=0;
for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      thrshld+=targabuffer[i+73][j]-erased[i][j];
   }
}
thrshld=thrshld/6270;

for(j=7; j<121; j++)
{
   for(i=0; i<55; i++)
   {
      if(thrshld<targabuffer[i+73][j]-erased[i][j])
         targabuffer[i+73][j]=0xff;
      else targabuffer[i+73][j]=0x00;
   }
   }

   for(j=0; j<7; j++)
   {
      for(i=0; i<55; i++)
      {
         targabuffer[i][j]=0x00;
         targabuffer[i][j+121]=0x00;
         targabuffer[i+73][j]=0x00;
         targabuffer[i+73][j+121]=0x00;
      }
   }


   /*write the binary file*/
   file1=fopen("\\targa\\lrt_comb.tga","wb");
   fwrite(targhead,18,1,file1);
   fwrite(targabuffer,1,16384,file1);
   fclose(file1);

   convert_targabuffer_to_temp();
   convert_temp_to_slm();
   write_to_slm();
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
prepare_slm()
{
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   *(semetex+0x800)=1;                                   /*clear slm*/
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_targabuffer_to_temp()
{
   for(i=0; i<128; i++)
   {
      for(j=0; j<128; j++)
      {
         if(targabuffer[i][j]==0x00) temp[i][j]=0x01;
         else temp[i][j]=0x00;
      }
   }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
convert_temp_to_slm()
{
   /*convert 2 pixels into an eight bit character for addressing the SLM*

   printf("creating SLM format  \n");
   for(x=0; x<128; x++)
```

```
    {
       for(j=0; j<16; j++)
       {
          slm[x][4*j]   =2*temp[126-8*j][x]+temp[127-8*j][x]+252;
          slm[x][4*j+1]=8*temp[124-8*j][x]+4*temp[125-8*j][x]+243;
          slm[x][4*j+2]=32*temp[122-8*j][x]+16*temp[123-8*j][x]+207;
          slm[x][4*j+3]=128*temp[120-8*j][x]+64*temp[121-8*j][x]+63;
       }
    }
}

/**  **  **  **  **  **  **  **  * * *  **  **  **  **  **  **  **  **/
write_to_slm()
{
   /*Write the SLM array to the SLM*/
   char far *semetex;
   semetex = (char far *) 0xb0000000;
   printf("writing to the SLM \n");
   for(j=0; j<16; j++)
   {
      for(x=0; x<128; x++)
      {
         i=x+128*j;
         *(semetex+i) = slm[x][4*j];
         *(semetex+i) = slm[x][1+4*j];
         *(semetex+i) = slm[x][2+4*j];
         *(semetex+i) = slm[x][3+4*j];
      }
   }
   *(semetex+0x801)=1;
}
/*************************************************************/
```

## BIBLIOGRAPHY

1. Goodman, J. W. *Introduction to Fourier Optics*. New York: McGraw-Hill Book Company, 1968.

2. Casasent, David and Demetri Psaltis. "Position, Rotation, and Scale Invariant Optical Correlation," *Applied Optics*, 15: 1795-1799 (July 1976).

3. Rogers, Steven K., and others. "Position, Scale, and Rotation Invariant Optical Pattern Recognition for Target Extraction and Identification," submitted to *Applied Optics* for publication in mid 1990.

4. Horev, Moshe. *Picture Correlation Model for Automatic Machine Recognition*. MS thesis, AFIT/GE/EE/80D-25. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1980 (AD-A100765).

5. Kobel, Capt William G. and Timothy Martin. *Distortion Invariant Pattern Recognition in Non-Random Noise*. MS thesis, AFIT/GE/ENG/86D-20. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986 (AD-A177598).

6. Troxel, 1st Lt Steven E. *Position, Scale, and Rotation Invariant Target Recognition Using Range Imagery*. MS thesis, AFIT/GEO/ENG/87D-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

7. Miazza, Capt David J., and others. "Optical In-Plane Distortion Invariant Pattern Recognition in Structured Noise," *Proceedings of the SPIE*, 939: 34-39 (April 1988).

8. Mayo, 2nd Lt Michael W. *Computer Generated Holograms and Magneto-Optic Spatial Light Modulator for Optical Pattern Recognition*. MS thesis, AFIT/GEO/ENG/87D-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

9. Childress, Capt Timothy G. and Capt J. Thomas Walrond. *Position, Scale, and Rotation Invariant Optical Pattern Recognition for Target Extraction and Identification*. MS thesis, AFIT/GE/ENG/88D [illegible]. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

10. Weaver, C. S. and Joseph W. Goodman. "A Technique for Optically Convolving Two Functions," Applied Optics, 5: 1248 (July 1966).

11. Javidi, Bahram, Don A. Gregory and Joseph L. Horner. "Single Modulator Joint Transform Correlator Architectures," Applied Optics, 28: 411 (February 1989).

12. Hill, Capt Marvin L., Feature Extraction Using Hough Transforms. MS Thesis, AFIT/GE/ENG/87D-24. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

13. Florence, James M. "Joint-transform Correlator Systems Using Deformable-mirror Spatial Light Modulators," Optics Letters, 14: 341-343 (April 1989).

14. "Signal Detection by Complex Spatial Filtering," IEEE Transactions on Information Theory IT-10, 139-145 (1964).

15. Javidi, Bahram. "Comparison of Bipolar Nonlinear Correlators and Phase-only Encoded Matched Filter Correlators," Proceedings of the SPIE, 956: 94-103 (1988).

16. Gregory, Don A., Jeffrey A. Loudin, and Francis T. S. Yu. "Illumination Dependence of the Joint Transform Correlation," Applied Optics, 28: 3288-3290 (August 1989).

17 Kabrisky, Dr. Mathew, Professor. Lecture notes taken in EENG 620, Pattern Recognition I. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1989.

18. Yu, Francis T. S., and others. "Effects of Fringe Binarization of Multiobject Joint Transform Correlation," Applied Optics, 28: 2988-2990 (August 1989).

19. Hughes, Capt Kenneth D. Suitability and Applications of Liquid Crystal Televisions In Optical Preprocessors. MS thesis, AFIT/CE/ENG/86D-6. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986.

20. Cederquist, J. and A. Tai. "Computer Generated Holograms for Geometric Transformations," Applied Optics, 23: 3099-3104 (September 1984).

21. Lee, Andrew J., Appendix to PhD dissertation, title unknown. Carnegie Mellon University PA (about 1986). Made available through Major Steven Rogers, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH.

22. Logue, James, Engineer. Telephone interviews. Electro-Optics Technology Division, Perkin-Elmer Corporation, Danbury CT, September 1989.

23. Zweig, David, Engineer. Telephone interview. Electro-Optics Technology Division, Perkin-Elmer Corporation, Danbury CT, September 1989.

24. Ayer, Capt Kevin, Gabor Transforms for Forward Looking Infrared Image Segmentation. MS thesis, AFIT/GEO/ENG/89D-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989.

25. Roggemann, Capt Michael C., Multiple Sensor Fusion for Detecting Targets in FLIR and Range Imagery. PhD Dissertation, AFIT/DS/ENP/89-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1989.

26. "Semetex Corporation SIGHT-MOD Development System Operations MAnual," Revised 30 November 1987. Semetex Corporation, Torrance CA.

27. Waas, Jay, Engineer. Telephone interview. Semetex Corporation, Torrance CA, November 1989.

28. "AT&T Truevision Advanced Raster Graphics Adapter TARGA 8 User's Guide," Release 1.1, 19 October 1985. AT&T Electronic Photography and Imaging Center, Indianapolis IN.

29. "Truevision TARGA Software Tools Notebook," Release 4.0, August 1988. Truevision Inc., Indianapolis IN.

30. "Truevision TARGA Demonstration" disk, 1986 release. AT&T Electronic Photography and Imaging Center, Indianapolis IN.

31. "TARGA Software Tools - Library Disk," Version 4.0, 19 May 1988. Truevision Inc., Indianapolis IN.

32. "Truevision Trutilities" disk, Version 1.1, 22 May 1989. Truevision Inc., Indianapolis IN.

33. "TARGA Software Tools - Utilities Disk," Version 4.0, 19 May 1988. Truevision Inc., Indianapolis IN.

34. "SONY CCD B/W Video Camera Module XC-38," XCM-38, 1986. Sony Corporation, Japan.

35. Ruck, Capt Dennis, PhD student. Personal interviews. Air Force Institute of Technology, Wright-Patterson AFB OH, November 1989.

36. "Spiricon 2250/2509 System Manual," Version 1.11, 7 December 1988. Spiricon Inc., Logan UT.

37. "Spiricon CQ-3D" disk, Version 5.0. Spiricon Inc., Logan UT.

38. "Series 150 Configuration Manual," 47-S15006-01. Imaging Technology Inc., Woburn MA.

39. Perez, Osvaldo, Engineer. Personal interviews. Aeronautical Systems Division (ASD\ENAML), Wright-Patterson AFB OH, January - November 1989.

40. Clay, Bruce, Engineer. Personal interview. Air Force Institute ot Technology (AFIT/EN), Wright-Patterson AFB OH, November 1989.

41. Childress, Capt Timothy, Engineer. Personal interviews. Foreign Technology Division (FTD), Wright-Patterson AFB OH, July - August 1989.

42. Rau, James E., "Detection of Differences in Real Distributions," Journal of the Optical Society of America, 56: 1490-1494 (November 1966).

43. Javidi, Bahram and Joseph Horner. "Single Spatial Light Modulator Joint Transform Correlator," Applied Optics, 28: 1027-1032 (March 1989).

44. Casasent, David and Demetri Psaltis. "New Optical Transforms for Pattern Recognition," Proceedings of the IEFE, vol. 65, 77-84 (January 1977).

45. Casasent, David and Others. "Real-Time Deformation Invariant Optical Pattern Recognition Using Coordinate Transformations, Applied Optics, 26: 938-942 (March 1987).

# VITA

Captain John D. Cline [redacted] attended Cuyahoga Community College until 1977 when he received Associate of Arts, and Associate of Applied Business degrees. He graduated from Cleveland State University in June 1978, with a Bachelors of Business Administration degree, and was employed as an accountant for the next two years, during which he passed the Certified Public Accountant examination. In January 1980, he served two years in the Merchant Marine. In January 1982 he returned to Cleveland State University, and graduated with the Bachelor of Science in Electrical Engineering in December 1984. During this time he was married to Terry Knupp (September 1982), and joined the United States Air Force (March 1984) under the College Senior Engineering Program. Upon graduation, he attended Officer Training School, and was commissioned in April 1985. He then served as team chief of a high frequency communication team with the 1815th Operational Test and Evaluation Squadron, Wright Patterson AFB OH until entering Air Force Institute of Technology.

[redacted]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
Distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GEO/ENG/89D-02

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Engineering

6b. OFFICE SYMBOL *(If applicable)*
AFIT/ENG

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Rome Air Development Cntr

8b. OFFICE SYMBOL *(If applicable)*
RADC/CA

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)*
Griffis AFB, NY 13441

10. SOURCE OF FUNDING NUMBERS

| PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO |
|---|---|---|---|
| | | | |

11. TITLE *(Include Security Classification)*
Hybrid Optical/Digital Architecture for Distortion Invariant Pattern Recognition

12. PERSONAL AUTHOR(S)
John D. Cline, Captain, USAF

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ________ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
1989 December

15. PAGE COUNT
291

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| 09 | 05 | |
| 12 | 09 | |

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Optical correlators; Pattern recognition

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
Thesis Advisor: Steven K. Rogers, MAJ, USAF
Associate Professor of Electrical Engineering

This research investigated optical techniques for pattern recognition. An optical joint transform correlator was implemented using a magneto-optic spatial light modulator, a charged coupled device (CCD) camera and framegrabber under personal computer (PC) control. A hybrid optical/digital architecture that could potentially perform position, scale, and rotation invariant pattern recognition using a computer generated hologram (CGH) was also implemented.

The joint transform correlator was tested using forward looking infrared (FLIR) imagery containing tactical targets, and gave very good results. New

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Steven K. Rogers, MAJ, USAF

22b. TELEPHONE *(Include Area Code)*
(513) 255-9266

22c. OFFICE SYMBOL
AFIT/ENG

*Previous editions are obsolete*

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

19 Continued:

techniques for binarizing the FLIR inputs and the fringe pattern of the joint transform were discovered. The input binarization used both the scene average and a localized energy normalization technique for binarization. This resulted in reduced scene background, while retaining target detail. The fringe binarization technique subtracted the Fourier transform of the scene from the joint transform, and binarized on the average difference. This new technique was a significant improvement over recent published designs.