SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... J H CROMER

UNCLASSIFIED DEC 82 AFIT/GE/EE/82D-26
SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

THESIS

APIT/GE/EE/82D-26  JAMES H. CROMER
2nd Lt      USAF

Approved for public release; distribution unlimited.
SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

THESIS

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

by
James H. Cromer, B.S.
2nd Lt USAF
Graduate Electrical Engineering
December 1982

Approved for public release; distribution unlimited.
Preface

One purpose of this study was to investigate a clutter-energy invariant target detection algorithm. A secondary purpose was to develop a base of image processing software for the AFIT Digital Signal Processing Laboratory NOVA-ECLIPSE minicomputer system, for use by future thesis students.

I would like to thank my advisor, Dr. Matthew Kabrisky, and committee members Major Larry Kizer and Major Kenneth Castor for their time and assistance during this study. I especially wish to express my gratitude to my wife, Karen, for her ceaseless support during my time at AFIT.

James H. Cromer
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>ix</td>
</tr>
<tr>
<td>Abstract</td>
<td>xi</td>
</tr>
<tr>
<td><strong>I. Introduction</strong></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>1</td>
</tr>
<tr>
<td>Justification</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>2</td>
</tr>
<tr>
<td>Problem/Scope</td>
<td>3</td>
</tr>
<tr>
<td>Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>Overview of Chapters</td>
<td>4</td>
</tr>
<tr>
<td><strong>II. Template Matching</strong></td>
<td></td>
</tr>
<tr>
<td>L1 and L2 Distance Measures</td>
<td>6</td>
</tr>
<tr>
<td>Normalized Cross-Correlation</td>
<td>8</td>
</tr>
<tr>
<td>Distance Factors</td>
<td>13</td>
</tr>
<tr>
<td><strong>III. Linear Processing</strong></td>
<td></td>
</tr>
<tr>
<td>Two-Dimensional Discrete Fourier Transform</td>
<td>16</td>
</tr>
<tr>
<td>Transform Theorems</td>
<td>18</td>
</tr>
<tr>
<td>Array Extension for Linear Correlation</td>
<td>20</td>
</tr>
<tr>
<td><strong>IV. Normalization Scheme</strong></td>
<td></td>
</tr>
<tr>
<td><strong>V. Software Description</strong></td>
<td></td>
</tr>
<tr>
<td>Image I/O</td>
<td>26</td>
</tr>
<tr>
<td>Scene and Template Synthesis</td>
<td>32</td>
</tr>
<tr>
<td>Correlation Implementation</td>
<td>34</td>
</tr>
<tr>
<td>Process Evaluation</td>
<td>38</td>
</tr>
<tr>
<td>Support Subroutines</td>
<td>40</td>
</tr>
<tr>
<td><strong>VI. Observations</strong></td>
<td></td>
</tr>
<tr>
<td>Clutter energy</td>
<td>43</td>
</tr>
<tr>
<td>Target Illumination</td>
<td>49</td>
</tr>
<tr>
<td>Positioning Errors</td>
<td>54</td>
</tr>
<tr>
<td>Classification Errors</td>
<td>58</td>
</tr>
<tr>
<td><strong>VII. Conclusion and Recommendations</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>63</td>
</tr>
<tr>
<td>Appendix A: Use of Macro Files</td>
<td>69</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Appendix B: Summary of Program Usage</td>
<td>73</td>
</tr>
<tr>
<td>Appendix C: Image I/O Programs</td>
<td>74</td>
</tr>
<tr>
<td>Appendix D: Scene and Template Synthesis</td>
<td>89</td>
</tr>
<tr>
<td>Appendix E: Correlation Implementation</td>
<td>114</td>
</tr>
<tr>
<td>Appendix F: Process Evaluation</td>
<td>126</td>
</tr>
<tr>
<td>Appendix G: Support Subroutines</td>
<td>148</td>
</tr>
<tr>
<td>Appendix H: Preliminary Results</td>
<td>154</td>
</tr>
<tr>
<td>Vita</td>
<td>156</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search window labelling convention.</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Indirect correlation method</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Array extension for linear correlation</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Normalization grid setup</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Scene PTANKD2, and it normalized by a 4x6 grid.</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Example of target sectoring problem</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>Equipment setup of the AFIT Digital Signal Processing Lab.</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>Packed and unpacked video formats</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>NMOVE input scene, and scene created by NMOVE.</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>Flowchart of scene synthesis</td>
<td>35</td>
</tr>
<tr>
<td>11</td>
<td>Flowchart of template synthesis</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>Flowchart of correlation implementation</td>
<td>39</td>
</tr>
<tr>
<td>13</td>
<td>Flowchart of evaluation process</td>
<td>41</td>
</tr>
<tr>
<td>14</td>
<td>Positive CCF for tank scene TEST.VD and template H3</td>
<td>44</td>
</tr>
<tr>
<td>15</td>
<td>Positive CCF for scene I3 and template H3</td>
<td>44</td>
</tr>
<tr>
<td>16</td>
<td>Negative CCF for tank scene TEST.VD and template H3, with Target-to-clutter area ratio of 0.12.</td>
<td>46</td>
</tr>
<tr>
<td>17</td>
<td>Negative CCF for scene I3 and template H3</td>
<td>46</td>
</tr>
<tr>
<td>18</td>
<td>Top 20% contour plot of Figure 17</td>
<td>47</td>
</tr>
<tr>
<td>19</td>
<td>Negative CCF for scene TEST.VD and template H3, with Target-to-clutter area ratio of 0.40.</td>
<td>48</td>
</tr>
<tr>
<td>20</td>
<td>Top 50% of negative CCF for scene I3 and template H3. Grid normalization used.</td>
<td>48</td>
</tr>
</tbody>
</table>
21 Digitized tank scene H3
22 CCF of PTANKH3 and PTEMPH3
23 CCF of NTANKH3 and NTEMPH3 (NPRODH3)
24 Top 30% contour plot of NPRODH3
25 Digitized tank scene D2
26 First two window choices from PEAK evaluation of NPRODD2
27 Histogram of template H3
28 Histogram of tank D2
29 Histogram of TONEDD2
30 Toned tank D2 and template H3
31 Suggested normalization scheme
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Distances from template H3 to a constant valued scene</td>
<td>13</td>
</tr>
<tr>
<td>II</td>
<td>Behavior of distance factors</td>
<td>15</td>
</tr>
<tr>
<td>III</td>
<td>PEAK evaluation of NPRODI3, and corresponding distances</td>
<td>50</td>
</tr>
<tr>
<td>IV</td>
<td>PEAK evaluation of NPRODH3, with corresponding distances</td>
<td>53</td>
</tr>
<tr>
<td>V</td>
<td>Distance scores from comparison of lower half of PTEMPH3 and PTANKH3</td>
<td>55</td>
</tr>
<tr>
<td>VI</td>
<td>PEAK evaluation of NPRODD2</td>
<td>57</td>
</tr>
<tr>
<td>VII</td>
<td>Distances corresponding to windows of Figure 26</td>
<td>57</td>
</tr>
<tr>
<td>VIII</td>
<td>Pixel mapping for toned scene D2</td>
<td>61</td>
</tr>
</tbody>
</table>
List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t(x,y)$</td>
<td>array representing the template</td>
</tr>
<tr>
<td>$s(x,y)$</td>
<td>array representing the scene</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>&quot;is defined as&quot;</td>
</tr>
<tr>
<td>$\sum(\cdot)$</td>
<td>summation of $(\cdot)$</td>
</tr>
<tr>
<td>$x \ast y$</td>
<td>region of overlap of scene and shifted template</td>
</tr>
<tr>
<td>$</td>
<td>\cdot</td>
</tr>
<tr>
<td>$D_1(m,n)$</td>
<td>distance metric based on L1 norm</td>
</tr>
<tr>
<td>$D_2(m,n)$</td>
<td>distance metric based on L2 norm</td>
</tr>
<tr>
<td>$t^T$</td>
<td>&quot;the vector $t$&quot;</td>
</tr>
<tr>
<td>$(\cdot)^T$</td>
<td>vector transpose of $(\cdot)$</td>
</tr>
<tr>
<td>$E_s(m,n)$</td>
<td>energy of scene search window centered at $(m,n)$</td>
</tr>
<tr>
<td>$E_t$</td>
<td>energy of template</td>
</tr>
<tr>
<td>$R_{st}(m,n)$</td>
<td>cross-correlation between scene and template</td>
</tr>
<tr>
<td>$N_{st}(m,n)$</td>
<td>normalized cross-correlation</td>
</tr>
<tr>
<td>$N_{L1}(m,n)$</td>
<td>normalized L1 distance function</td>
</tr>
<tr>
<td>$N_{L2}(m,n)$</td>
<td>normalized L2 distance function</td>
</tr>
<tr>
<td>$S_{L1E}$</td>
<td>L1 energy of scene</td>
</tr>
<tr>
<td>$T_{L1E}$</td>
<td>L1 energy of template</td>
</tr>
<tr>
<td>$C_{FACTOR}$</td>
<td>correlation distance factor</td>
</tr>
<tr>
<td>$L_{2FACTOR}$</td>
<td>L2 distance factor</td>
</tr>
<tr>
<td>$L_{1FACTOR}$</td>
<td>L1 distance factor</td>
</tr>
<tr>
<td>$\tilde{g}(x,y)$</td>
<td>periodic extension of $g(x,y)$</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>indicates Fourier Transform pair</td>
</tr>
<tr>
<td>$F{(\cdot),}$</td>
<td>Fourier Transform $(\cdot)$</td>
</tr>
</tbody>
</table>
\[ F \{\cdot\}\] -- inverse Fourier Transform of \(\cdot\)

\( T^* \) -- complex conjugate of \( T \)

\((H_i,V_j)\) -- grid rectangle in grid row \( j \), grid column \( i \)

\( s'(x,y) \) -- normalized array

\( N_{ij} \) -- normalized coefficient

\( E_{ij} \) -- energy of grid rectangle \((H_i,V_j)\)

\( N_g(m,n) \) -- statistical correlation measure
This work focuses on a method for two-dimensional pattern recognition. The method includes a global search scheme for candidate windows of interest, based on Fourier domain cross-correlation. A method to normalize the input scene by local rectangular regions, in an attempt to efficiently approximate search window normalization, is presented. Also developed is a candidate window (potential target) similarity measure, based on the normalized L1 and Euclidean distances, which is independent of the template DC value and its energy. Observations on the performance of the algorithm applied to visual spectrum photographs of tanks in a realistic environment are included. Also included is the software needed to implement the algorithm on a Data General Eclipse S/250 minicomputer.
SCENE ANALYSIS: NON-LINEAR SPATIAL FILTERING FOR AUTOMATIC TARGET DETECTION

I. INTRODUCTION

GENERAL

Target detection is the area of pattern recognition concerned with locating a given class of objects embedded within a background scene. The distinction to be made is whether the object in question belongs in the target class or in the non-target class. The challenge of this type of automatic analysis of complex visual data by machine has proven to be surprisingly difficult. The problem remains largely unsolved despite considerable research effort [1: 28].

JUSTIFICATION

Automated systems capable of identifying objects in a cluttered background, irrespective of size, orientation, illumination, or position would have a nearly unlimited range of applications. The following list suggests a few of the major areas [2: 596]:

1) Document processing: recognition of unformatted, non-segmented, multi-font characters;

2) Industrial automation: robot assembly and inspection;

3) Military applications: analysis of reconnaissance imagery, enhancement of weapon delivery display systems, and realization of the autonomous missile.
BACKGROUND

The target detection process can be broken down into three steps as follows:

1) Determine characteristic features of the target template that are invariant to size, orientation, energy, and background changes;

2) Perform a global search of the cluttered input scene for the features of step (1) and identify local "windows" of interest (target candidates); and

3) Classify the information within the candidate windows into one of the two disjoint classes of either targets or non-targets.

The problems associated with this target detection process are numerous. One problem is in analytically describing the quintessence of a class of objects, given the physical characteristics of only a finite number of template objects. Once a suitable set of target characteristics (or features) is selected, the isolation of those features from a background scene which may possess some of the same features as the target may be impossible. For these and other reasons, no general solution exists to detect objects embedded in "real-life" cluttered scenes.

Previous studies performed in this area at the Air Force Institute of Technology have shown promising, although limited, results. One of these studies is a thesis by
Israeli Air Force Major Moshe Horev, "Picture Correlation Model for Automatic Machine Recognition" [3]. In it, Horev describes a series of transformations performed in the Fourier domain to determine the size and orientation of targets within a cluttered scene. He then suggests to perform a non-linear (and, hence potentially unpredictable) operation of combining the modified phase of the scene with the magnitude of the template in hopes of enhancing the target objects; this procedure is known as the "Phase of the Image, Magnitude of the Template" (PIMT) process. An immediate observation is that the success of the process may be both scene and template dependent. The PIMT process has debatable merit, but the existence of a scale-rotation transformation does suggest an area for further research. With the premise that the size and orientation of a candidate target within a scene is known or can be determined, can a process be developed to then locate and accurately classify the target?

PROBLEM/SCOPE

This study includes:

1) The initial development of a target detection algorithm that is invariant to background scene composition or energy;

2) The implementation on a digital minicomputer of the software routines needed to process the input imagery, perform global scene searches through high-speed correlation,
and discriminate local candidate windows by using L1 and L2 distance metrics; and

3) The preliminary test results. A complete statistical performance analysis of the process was not conducted due to time constraints. Suggestions for improving the performance of the process are included.

ASSUMPTIONS

The following assumptions were made concerning the input test scenes:

1) The size and orientation of targets in a scene is known, or can be determined by existing methods, three of which are diffraction pattern sampling, cross-correlation with a bank of scaled and rotated templates, or by performing a scale-rotation transformation;

2) The digitized scene images are accurate representations of the continuous scenes from which they were obtained;

3) The digitized scene images may have been corrupted by additive uncorrelated noise; and

4) Illumination over the continuous scene is varying slowly.

OVERVIEW OF PRESENTATION

The next chapter discusses methods of detecting objects in scenes through the classic technique of template matching. Distance factors based on the L1 and L2 distance metrics are
derived; these factors will be used to classify candidate windows, or potential targets.

Chapter three briefly discusses the discrete Fourier transform and some of its properties. A method for performing high-speed correlation by multiplication in the Fourier domain is given; cross-correlation will be used in the detection process to perform searches for potential targets in the input scene.

In chapter four a scene normalization scheme is presented. The normalization is necessary to improve the ability of the cross-correlation to locate candidate targets.

The software needed to implement the detection process is described in chapter five. The source code has been included in the appendix.

In chapter six, some observations of detection process are made. Explanations are given for the weaknesses of the algorithm, and suggestions for improving the performance are discussed.
II. TEMPLATE MATCHING

Often in scene analysis problems a simple question is to be answered: Does the input scene contain a previously specified object? A technique classically employed to determine the presence of an object is the fundamental method of template matching, in which the template brightness function is compared point-by-point with the scene brightness function. In most cases, a perfect template match will not be found, so some realistic distance measure \( D(m,n) \) indicating the degree of similarity between the template window and the scene needs to be computed for all possible points in the scene.

**L1 and L2 DISTANCE MEASURES**

Let the array (or vector) \( t(x,y) \) represent in some sense the template pattern, and let the array \( s(x,y) \) represent the scene to be searched. For our purposes of discussion, it is immaterial how the arrays are obtained, whether from digitizing (sampling and quantizing) the continuous brightness distributions, infrared distributions, or some function of these distributions (for example only the low-frequency Fourier components). Two common definitions of distance measures used are given by equations (1) and (3) [4: 279].

\[
D_{1}(m,n) \triangleq \sum_{x} \sum_{y} |s(x,y) - t(x-m,y-n)| \tag{1}
\]

\[
= \sum_{x} \sum_{y} \sqrt{(s(x,y) - t(x-m,y-n))^{2}} \tag{2}
\]
\[ D_2(m,n) = \sqrt{\sum_x \sum_y [s(x,y) - t(x-m,y-n)]^2} \] (3)

* for all \( x, y \) such that \( (x-m,y-n) \) is within the area of overlap of the scene and template windows \((0 < m < M+J, 0 < n < N+K \text{ for a } JxK \text{ template and an } MxN \text{ search area})\). See Figure 1 for an illustration of the labelling convention used. Note that \( m \) and \( n \) represent a specific translation between \( s(x,y) \) and \( t(x,y) \).
The definitions for D1(m,n) and D2(m,n) are known as the metrics based on the L1 and L2 norms respectively. The D2(m,n) measure is also known as the standard Euclidean distance between two vectors, that is

\[
D2(m,n) = (s - \hat{t})^T (s - \hat{t})
\]

where

\[
\hat{t} = \begin{bmatrix}
t(x_1-m, y_1-n) \\
t(x_2-m, y_2-n) \\
\vdots \\
t(x_n-m, y_n-n)
\end{bmatrix}
\]

NORMALIZED CROSS-CORRELATION

Insight can be gained from equation (3) by expanding it as follows:

\[
D2(m,n)^2 = \sum_x \sum_y \left[ s^2(x,y) - 2s(x,y)t(x-m,y-n) + t^2(x-m,y-n) \right]
\]

Equivalently,

\[
D2(m,n)^2 = \sum_x \sum_y s^2(x,y) - 2 \sum_x \sum_y s(x,y)t(x-m,y-n) + \sum_x \sum_y t^2(x-m,y-n)
\]

Let Es, Et, and Rst be defined as follows:

\[
Es(m,n) \triangleq \sum_x \sum_y s^2(x,y)
\]

\[
Et(m,n) \triangleq \sum_x \sum_y t^2(x-m,y-n)
\]
\[ R_{st}(m,n) = \sum_x \sum_y s(x,y) t(x-m,y-n) \] (9)

(There is no restriction on the range of \( x \) and \( y \) in equations (8) and (9) as the template function is considered to be zero outside the area of interest.)

The term \( E_s(m,n) \) represents the scene energy (or equivalently the vector length) of the search window, which will vary over the search area. The term \( E_t(m,n) \) represents the template energy, which is constant for all values of \( m \) and \( n \). The term \( R_{st}(m,n) \) is the cross-correlation between the scene and template, and generally is largest when the distance \( D_2(m,n) \) is smallest. The cross-correlation term is not an absolute measure of the template difference, however, since the scene window energy \( E_s(m,n) \) is position variant. For this reason, \( R_{st}(m,n) \) is normalized to achieve invariance to input energy [5: 553]. The normalized cross-correlation, denoted by \( N_{st}(m,n) \), is defined as follows:

\[
N_{st}(m,n) \triangleq \frac{R_{st}(m,n)}{\sqrt{E_t(m,n)} \sqrt{E_s(m,n)}}
\] (10)

where the usual restriction is placed on \( x \) and \( y \) for the computation of the scene energy term.

The significance of this normalization can be realized by appealing to the Schwarz inequality [6: 159]. The Schwarz
inequality is stated as follows:

For two real functions $f(x)$ and $g(x)$ defined on $a < x < b$,

$$\sum f(x)g(x) \leq \sqrt{\sum[f(x)]^2} \sqrt{\sum[g(x)]^2}$$  \hspace{1cm} (11)

with equality when $f(x) = kg(x)$, where $k$ is a constant scale factor. As applied to the cross-correlation terms,

$$\sum \sum s(x,y)t(x-m,y-n) \leq \sqrt{\left[\sum \sum s^2(x,y)\right]} \sqrt{\left[\sum \sum t^2(x-m,y-n)\right]}$$  \hspace{1cm} (12)

or, by rearranging terms

$$\text{Nst}(m,n) \leq 1$$  \hspace{1cm} (13)

with equality only when the scene area under consideration exactly matches the template. Thus the normalized cross-correlation can be used as a decision criterion regardless of the distribution of the non-normalized scene energy, assuming that the scene energy can be recomputed for each shift of the search window. A simple decision rule would classify the scene window information into the target class only when Nst exceeded some preset upper threshold value, into the non-target class when Nst was below a lower threshold, and would not make a decision when Nst was between the thresholds.

With the constraint that the scene and template can take on only non-negative values, a lower bound for the threshold value is zero. Fortunately, a tighter bound closer to unity
can be determined for cross-correlations with scenes of interest (ones that approach the template in form). Consider a scene of constant value $c$, $c > 0$. Then the cross-correlation value may be determined by applying equations (7) through (10):

$$E_s = \sum_x \sum_y c^2$$

(14)

$$E_s = c \frac{2}{JK}$$

(15)

$$[Nst|s=c] = \frac{c \sum_x \sum_y t(x-m,y-n)}{c \sqrt{JK} \sqrt{Et}}$$

(16)

$$Nst|c = \frac{\sum_x \sum_y t(x-m,y-n)}{\sqrt{EtJK}}$$

(17)

with $J = \text{width of search window}$

$K = \text{length of search window}$

Note that the normalized cross-correlation between the template and a scene of constant non-zero value, $Nst|c$, is independent of the scene value. Scene windows which yield a $Nst$ less than $Nst|c$ need not be considered for further discrimination.

**NORMALIZED L1 AND L2 DISTANCES**

Two other distance measures which take into account the array energies are the normalized L1 and L2 distance measures, defined in Eqs. (18) and (19).
The normalized L2 distance can be determined more efficiently by Eq. (22):

\[
NL2(m,n) = \sqrt{2 \left[ 1 - Nst(m,n) \right]}
\]  

One way of "visualizing" the normalized L2 distance is to think of it as the Euclidean distance between the points where the scene vector and the template vector intersect the unit hypersphere. Thus NL2 is dependent only upon the angle between the vectors, and not on the vector lengths.

The maximum normalized distances to be considered as possibly identifying a target location will be those corresponding to the distances computed between a template and a scene with a constant value. These distances are given in Table I. for typical tank template.
**TABLE I. DISTANCES FROM TEMPLATE H3 TO A CONSTANT VALUED SCENE**

<table>
<thead>
<tr>
<th>CENTER (ROW,COLUMN)</th>
<th>TOP ROW COLUMN</th>
<th>EUCLIDEAN DISTANCES</th>
<th>NORMALIZED EUCLIDEAN</th>
<th>L1 DISTANCE</th>
<th>NORMALIZED L1</th>
<th>NORMALIZED CORRELATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>112,44</td>
<td>90</td>
<td>641</td>
<td>90b1</td>
<td>39509</td>
<td>468</td>
<td>872</td>
</tr>
</tbody>
</table>

**COMMENT** - WHITE TD HAS A CONSTANT VALUE OF 15

**DISTANCE FACTORS**

At this point the distance factors used to classify candidate target windows will be introduced. The correlation, L1, and L2 factors correspond to the normalized correlation, L1, and L2 distances linearly scaled into a 0-100 range, with 100 corresponding to an exact match and 0 corresponding to the distance to a constant valued scene. Consider the mappings

\[
\begin{align*}
Nst & \mapsto 0 \quad \text{for } Nst \leq Nst|c \\
NL2 & \mapsto 0 \quad \text{for } NL2 \geq NL2|c \\
NL1 & \mapsto 0 \quad \text{for } NL1 \geq NL1|c \\
\end{align*}
\]

and

\[
\begin{align*}
Nst & = 1 \mapsto 100 \\
NL2 & = 0 \mapsto 100 \\
NL1 & = 0 \mapsto 100
\end{align*}
\]
The functions to achieve these mappings are

\[
\text{CFACtor} = \begin{cases} 
100[(Nst - Nst|c|)/(1 - Nst|c|)] & \text{Nst} > \text{Nst}|c| \\
0 & \text{else}
\end{cases} \quad (25)
\]

\[
\text{L2FACTOR} = \begin{cases} 
100[1 - \text{NL2}/\text{NL2}|c|] & \text{NL2} < \text{NL2}|c| \\
0 & \text{else}
\end{cases} \quad (26)
\]

\[
\text{L1FACTOR} = \begin{cases} 
100[1 - \text{NL1}/\text{NL1}|c|] & \text{NL1} < \text{NL1}|c| \\
0 & \text{else}
\end{cases} \quad (27)
\]

A score is computed to take into account all three distance measures as follows:

\[
\text{SCORE} = \sqrt[3]{(\text{CFACtor} \times \text{L2FACTOR} \times \text{L1FACTOR})} \quad (28)
\]

Note that SCORE intentionally favors the Euclidean measure over the L1 metric (recall that the normalized correlation is an invertible function of the normalized Euclidean distance). The SCORE will always be a number from 0 to 100 inclusive. The behavior of the SCORE of a template measured against itself for various window shifts is given in Table II.

One of the problems in determining the distance functions is that they are computationally expensive, often infeasible, because of the large size arrays required for most applications. This is the case with many linear
processing algorithms. Indirect computational techniques based on unitary transforms permit more efficient linear processing than conventional methods. The next chapter introduces the Fourier transform as a linear processing tool.
An efficient method of linear processing is through the use of unitary transforms. A unitary transform meets the following three conditions [5: 232]:

1) It is a linear transformation; 
2) Its operation is exactly invertible; and 
3) Its operating kernel satisfies certain orthogonality conditions.

A unitary transform of particular importance in the field of image processing is the two-dimensional Fourier transform. In addition to its use as a linear processing tool, the Fourier transform provides a means of extracting features from images. For instance, the center or DC term is proportional to the average image brightness. The low-frequency terms contain the gross form information, while the high-frequency terms indicate the amplitude and orientation of the edges (the detail).

**TWO-DIMENSIONAL DISCRETE FOURIER TRANSFORM**

Consider a two-dimensional periodic sequence

\[ \tilde{g}(x,y) = g(x+qM, y+rN) \]  

(29)

where \( q \) and \( r \) are integers, and \( M \) and \( N \) are the periods in the \( x \) and \( y \) direction. Such a sequence can be represented by a finite sum of exponentials in the form

\[ \tilde{g}(x,y) = \frac{1}{MN} \sum_{fx=0}^{M-1} \sum_{fy=0}^{N-1} \hat{g}(fx, fy) \exp[j2\pi(xfx/M+yfy/N)] \]  

(30)
where
\[ \tilde{g}(f_x, f_y) = \sum_x \sum_y \hat{g}(x, y) \exp[-j2\pi(xf_x/M + yf_y/N)] \]  \hspace{1cm} (31)

and \( j = \sqrt{-1} \).

Note that \( \tilde{g}(f_x, f_y) \) will have the same periodicity as the sequence \( \hat{g}(x, y) \). If a finite area sequence \( g(x, y) \) is considered to be one period of \( \hat{g}(x, y) \), and \( G(f_x, f_y) \) is taken to be one period of \( \tilde{g}(f_x, f_y) \), then \( g(x, y) \) and \( G(f_x, f_y) \) will form a discrete Fourier transform pair. In equation form [7: 117],

\[
g(x, y) = \begin{cases} 
\frac{1}{MN} \sum_{f_x} \sum_{f_y} G(f_x, f_y) \exp[j2\pi(xf_x/M + yf_y/N)] & 0 \leq x \leq M-1 \\
0 & \text{otherwise}
\end{cases} \hspace{1cm} (32)
\]

\[
G(f_x, f_y) = \begin{cases} 
\sum_{x} \sum_{y} g(x, y) \exp[-j2\pi(xf_x/M + yf_y/N)] & 0 \leq f_x \leq M-1 \\
0 & \text{otherwise}
\end{cases} \hspace{1cm} (33)
\]

The notation to be used to indicate a Fourier Transform pair is

\( g(x, y) \leftrightarrow G(f_x, f_y) \) \hspace{1cm} (34)

Equivalently,

\[ F\{g(x, y)\} = G(f_x, f_y) \] \hspace{1cm} (35)

and

\[ F^{-1}\{G(f_x, f_y)\} = g(x, y) \] \hspace{1cm} (36)
Having defined the Fourier transform, two theorems for use in later developments will be stated without proof [7: 110].

**SHIFT THEOREM**

If \( t(x, y) \rightarrow T(f_x, f_y) \),
then \( t(x-m, y-n) \rightarrow \exp[-j2\pi(mf_x/M+nf_y/N)]T(f_x, f_y) \) \( (37) \)

**REVERSAL THEOREM**

If \( t(x, y) \rightarrow T(f_x, f_y) \),
then \( t(-x, -y) \rightarrow T^*(f_x, f_y) \),
where \( T^* \) is the complex conjugate of \( T \). \( (38) \)

The convolution theorem suggests a method for performing correlation in the Fourier domain. It will now be stated with its proof, modeled after a proof for continuous signals [4: 307].

**CONVOLUTION THEOREM**

For \( s(x, y) \rightarrow S(f_x, f_y) \) and \( t(x, y) \rightarrow T(f_x, f_y) \),

\[ F\{s(x, y)*t(x, y)\} = S(f_x, f_y)T(f_x, f_y) \] \( (39) \)

**Proof:**

By definition,

\[ F\{ \sum_x \sum_y s(x, y)t(m-x, n-y) \} = \] \( (40) \)

\[ \sum_m \sum_n [\sum_x \sum_y s(x, y)t(m-x, n-y)]\exp[-j2\pi(mf_x/M+nf_y/N)] \]

Interchanging the summation order,

\[ = \sum_x \sum_y s(x, y)\{ \sum_m \sum_n t(m-x, n-y)\exp[-j2\pi(mf_x/M+nf_y/N)] \} \]
The same template is correlated with many different inputs, discrete Fourier transforms. Note that in many applications, the correlation of two sequences is given by the use of the discrete Fourier transform of a finite-duration sequence. Thus, a computationally reasonable implementation of the discrete Fourier transform of a finite-duration sequence results in algorithms exist for computing the highly efficient algorithms exist for computing the

\[ (\mathcal{F}x', x) \mathcal{L}(\mathcal{F}x, x)S \]  

By the reverse theorem it follows that

\[ (\mathcal{F}x', x) \mathcal{L}(\mathcal{F}x', x)S = \]  

\[ \{(\mathcal{F}x', x) \mathcal{L}\left[(\mathcal{F}x' + \mathcal{F}x, x) \mathcal{L}^{-1}\{\mathcal{F}xS\}\right] \} \]  

Application of the shift theorem gives
scenes, so that \( T^*(fx, fy) \) needs to be computed just once and stored.

**ARRAY EXTENSION FOR LINEAR CORRELATION**

Care must be taken in choosing sequence and transform lengths. Consider the linear correlation between two N-point sequences

\[ R(m) = \sum s(x) t(x-m) , \]

where \( R(m) \) will have up to 2N-1 non-zero points. The indirect correlation method using N-point discrete Fourier transforms will result in an N-point sequence, which is the circular correlation of the input sequences. To obtain the linear correlation, the discrete Fourier transforms must be computed on the basis of 2N-1 or more points, with the input sequences extended with at least N-1 zeros. In general, for \( s(x) \) of length \( S_1 \) and \( t(x) \) of length \( T_1 \), the indirect linear correlation method requires that discrete Fourier transform be computed on the basis of at least \( S_1 + T_1 - 1 \) points.

For the two-dimensional case, the sequence arrays are extended as follows (See Figure 3) [5: 288]:

1) Imbed the \( T_1 \times T_2 \) template image sequence in the lower right quadrant of an all zero \( M_1 \times M_2 \) matrix;

2) Imbed the \( S_1 \times S_2 \) input scene in the upper left quadrant of an all zero \( M_1 \times M_2 \) matrix;

3) Compute all discrete Fourier transforms on the basis of \( M_1 > S_1 + T_1 - 1 \) and \( M_2 > S_2 + T_2 - 1 \) to avoid wrap-around error.
By taking advantage of the speed gained by implementing the indirect correlation method, the ability to normalize the scene search window during the correlation process is lost. This inability is a major shortcoming of the indirect correlation method. An alternate normalization scheme must be implemented to approximate the window-by-window normalization method.
Consider dividing the scene to be normalized into a grid of rectangles. A compromise between global normalization and search window normalization would be to divide each of the scene values within a given rectangle by some constant that is proportional to the square root of the energy of that rectangle. This normalization scheme may be implemented as in Figure 4.

\[ (H_i, V_j) \quad \text{-- grid rectangle in grid row } j, \text{ grid column } i \]

\[ N \quad \text{-- of pixel rows} \]

\[ M \quad \text{-- of pixel columns} \]

\[ (i, j, M, N, N/k, \text{ and } M/l \text{ are all integers}) \]
Define

\[ E_{ij} \triangleq \sum_{x} \sum_{y} s^2(x, y) \tag{46} \]

for \((i-1)M/1 + 1 \leq x \leq iM/1\)

and \((j-1)N/k + 1 \leq y \leq jN/k\)

Then for

\[ s'(x, y) \quad \text{-- normalized array value} \]

\[ N_{ij} \quad \text{-- normalization coefficient} \]

\[ E_{ij} \quad \text{-- energy in grid rectangle \((Hi,Vj)\)} \]

\[ s'(x, y) \triangleq \frac{s(x, y)}{N_{ij}} \tag{47} \]

where

\[ N_{ij} = \sqrt{E_{ij}} \tag{48} \]

Some thought must be given to the size of the rectangles chosen, for as the size is increased, the clutter energy-to-target energy ratio is also increased (ideally this ratio should be zero). As the size of the rectangles is decreased, the scene begins to lose contrast as the normalized values asymptotically approach a constant value (namely 1 when the rectangle size is 1x1). Another problem accompanying the decrease in rectangle size is the possibility of sectoring part of a target into separate rectangles increases; this could have a deleterious effect on the cross-correlation function. See Figure 5 for an illustration of a normalized scene.
FIGURE 5: TOP: SCENE PTANED2.  
BOTTOM: SCENE NORMALIZED WITH A 4X6 GRID.
Consider the following two cases, shown in Figure 6, in which a 3x6 normalization grid has been chosen.

**FIGURE 6: EXAMPLE OF TARGET SECTORING PROBLEM.**

In case I, the target will be satisfactorily normalized against the high energy background, allowing a successful global search by correlation. In case II, the clutter-to-target area is large for all grid rectangles. The high energy areas dominate, yielding false peaks in the correlation function.

In the next chapter, the software necessary to process the images and test the target detection process is described. Also, the detection process is further discussed.
V. SOFTWARE DESCRIPTION

The software that was used in this study is described in this chapter. The programs (indicated by capitalized titles) have been grouped into the following categories:

1) Image input and output;
2) Scene and template synthesis;
3) Correlation implementation;
4) Process evaluation; and
5) Support subroutines.

All source code that was generated during this thesis effort and added to the AFIT Signal Processing Laboratory software archives is included in the appendix. All programs are written in Data General (DG) Fortran 5 (except for VIDEO7 and NMOVE, which are written in Fortran IV). All programs were written by James Cromer, with the exception of PLTTRNS, INVERSE, and DIRECT (by Ronald Schafer), and IOF, UNPACK, and REPACK (by Robin Simmons).

IMAGE I/O

Before a digital computer can be used to analyze an image, the image must first be converted to a form usable by the computer. Specifically, the image must be represented in some sense by an array of numbers. The process used to obtain this array is known as digitization, in which some image parameter is sampled and quantized at points throughout the scene. In this study, the achromatic brightness, or gray
level, is sampled in a 256 column by 256 row quadruled grid format, and quantized into one of 16 levels (4-bit digitization). The resulting array values are referred to as "pixels", short for "picture elements."

The equipment used in the process include a standard video monitor, a Cohu 6150 vidicon camera with 6950 camera controls, and 3 Tecmar digitizer boards (A/D converter, direct memory access, D/A). The Tecmar digitizer is interfaced with a DG NOVA 2 processor via a CROMEMCO Z-80 based microcomputer. The NOVA terminal can be used to communicate with the A/D/A converter with the Fortran callable subroutine CHANNEL, developed in an earlier AFIT thesis [8]. High speed processing can be performed on the digitized images with the powerful DG ECLIPSE S/250 minicomputer. Both DG machines are 16-bit processors. See Figure 7 for a schematic of the equipment layout.

The program which controls image input (digitization) and output (display) through CHANNEL is VIDEO7. When running VIDEO7 and the input option is chosen, seven digitized versions of an input image are stored in files named "A0" through "A6", which can be averaged together later. When the output option is chosen, the user is given the choice to display from one to ten files named "A0" through "A(n-1)", where n is the number of files to be displayed. In addition to the main purpose of checking images before averaging, n
Figure 7: Equipment Layout of the AFIT Signal Processing Laboratory
images of interest may be easily presented in sequence for
demonstrations by re-naming the images "A0" through "A(n-1)."
The third mode of VIDEO7 allows the user to display an
existing file any number of times consecutively. This mode
is used when the D/A converter malfunctions, and usually
several attempts to view an image must be made before a
satisfactory image is displayed.

Files created by VIDEO7 are written to disk in what is
referred to as "packed video form." The file is "packed"
because 4 pixels are stored in each 16-bit word. Packed form
is ideal for minimizing storage requirements, while posing
only minor processing inconveniences. In packed form, video
files will be 64 blocks long, where one block is 256 16-bit
words. Thus one image requires only 32K bytes of memory.
Note that each block holds 4 packed video rows. As a result,
the processing programs operate on multiples of 4 rows at a
time between RDBLK calls. The fastest way to transfer data
from disk storage to core memory is by the RDBLK call, which
passes data in the data channel mode. The data channel mode
of moving data does not require program control once a
transfer is initiated. Figure 8 shows more clearly the
relation between packed and unpacked forms.

The seven digitized images created by VIDEO7 can be
averaged to produce an output image that has an improved
signal to noise ratio. The program that does this averaging
### Bit Position Numbering Convention

**Used by ISET and ITEST.**

**Figure 8: Packed and Unpacked Video Formats.**
is called QUICKAVE7, which creates an output file named "AVERAGE7.VD", where the .VD extension indicates a video file. The seven files to be averaged are assumed to be named "A0" through "A6."

It is often desirable to produce a hard copy of a digitized image. One of the ways this can be done is by displaying an image through VIDEO7, then activating the Tektronix 4632 Video Hard Copy Unit, which makes a photocopy of the image being sent to the video monitor. This method is acceptable most of the time, but for numerous reasons it is necessary to produce hard copies of stored images with the Printronix P-300 lineprinter. The program originally written to do this is DISPLAY, by Robin Simmons in an earlier thesis [9]. DISPLAY used 3x3 dot patterns to simulate the 16 gray-levels, which resulted in two shortcomings. Distortion occurs in the picture because the P-300 horizontal dot density is less than the vertical density, resulting in a 1.2:1 aspect ratio. Also the 3x3 patterns do not fully take advantage of the 16 gray-levels available. DISPLAY was modified to solve these problems by using a combination of 3x3 and 3x4 dot patterns. Up to four different dot patterns are used per gray-level, instead of just one, and the aspect ratio is very nearly 1:1. Other modifications include allowing the user to choose the number of rows to be displayed along with the starting row. The run time for an 11x13 image hard copy was reduced to less than 90 seconds,
down from 5-6 minutes. The modified program is called PICTURE.

SCENE and TEMPLATE SYNTHESIS

After an image is digitized and stored, the next step is to create a template or scene to be used in the correlation process. To do this programs were written to improve the image, create a scene or template by combining images, and to put the images into the correct format for the correlation process.

The program REMOVE was written to perform a 3x3 pixel mask processing of an image for the purpose of noise removal. The main program handles the bookkeeping of passing the three rows to be operated on to the subroutine TEST3, which produces the noise-removed output row. The subroutine UNPACK2 is used to unpack the video rows from four pixels per word to one pixel per word. REMOVE was not used extensively, but is included to demonstrate an efficient method to perform mask processing. The mask function can be changed by modifying TEST3. TEST3 presently computes the difference between the center pixel value and the average value of the surrounding pixels. If the difference is greater than some threshold, the center pixel value is modified accordingly.

If REMOVE and QUIKAVE7 fail to produce a satisfactory image, a histogram can be generated, and then modified to enhance the image. The program to produce the histogram is
called EVIDHIST (the "E" indicates an Eclipse only program). TONER modifies the histogram by a mapping function of the type: \(0 \rightarrow a, 1 \rightarrow b, \ldots, 15 \rightarrow p\), where the user defines the new values of "a" through "p". TONER is used to increase the contrast or raise the average brightness level when deficiencies occur due to A/D or camera gain misadjustment.

To create a template from a scene with a target in it, a "window" is placed over the target information, and the background is set to some constant value (usually 0 or 15).

Program NMOVE allows the user to specify a template scene file, a background file, and a combined filename. The template window size and position are variable, as is the combined window position. Figure 9 demonstrates the capability of NMOVE.

![Figure 9: LEFT: NMOVE INPUT SCENE. RIGHT: SCENE CREATED BY NMOVE. THE BACKGROUND SCENE IS THE 16 GRAY-LEVEL BARPATTERN.](image)
The next step requires that the "negative" image be formed, using the equation \[ \text{NEGATIVE} = 15 - \text{POSITIVE}. \] This forces the expected high energy background to become a low energy background, improving the correlation results. TONER can also be used for this purpose, but NEGATE is used in macro files, as it requires no user input.

The last step before the correlation sequence is to do a four-to-one reduction to imbed the 256x256 scene into the upper left quadrant (lower right for template) so that linear correlation will be obtained. Program REDUCE does this by averaging four pixels to create one output value. This reduction is also effectively a low-pass filtering operation. See Figures 10 and 11 for flowcharts of the scene and template synthesis processes.

**CORRELATION IMPLEMENTATION**

The correlation method used is to perform a multiplication in the frequency domain, then inverse transform the product, resulting in a correlation in the spatial domain. The complex arrays multiplied have been obtained from Fourier transforming normalized image arrays.

The normalization scheme used is the rectangle grid normalization described in Chapter 4. The program which carries out the normalization is suitably named NORMALIZE. The number of grid rows and grid columns are chosen by the user. Only the upper left or lower right quadrants are
FIGURE 10: FLOWCHART OF SCENE SYNTHESIS.
FIGURE 11: FLOWCHART OF TEMPLATE SYNTHESIS.
normalized; the remainder of the array is set to zero. The input file is a 256x256 packed video, and the output is a 256x256 complex file.

After the normalized scene and template complex files have been created, they are Fourier transformed by the program DIRECT. The resultant arrays are then complex multiplied by CMULTIPLY, and the product is inverse transformed by INVERSE. INVERSE and DIRECT are Eklundh FFT algorithm based programs written by Ron Schafer.

The file created by INVERSE is a complex array with the imaginary part zero (since the scene and template arrays are always real), and with the real part having values between zero and two. CTOI conserves file space by converting the complex array to integer form by multiplying each real number by 16384 to take advantage of the 16 bit word. The integer file uses only one-fourth as much disk memory as a complex file.

The program CTOV can be used to convert a complex file (imaginary part assumed zero) into a video file. It performs a linear scaling of the input file into a 0-15 range. CTOV was used (along with PICTURE) to display the complex normalized scene of Figure 5.
The last step in the correlation process is to combine the results of several correlations between a scene and a set of templates. Program IMULTIPLY computes the geometric mean of two correlation functions by determining the square root of the product of the input arrays. An example of when IMULTIPLY may be used is when the correlation functions created from the left-half and the right-half of a template are to be combined. IMULTIPLY can also be used to combine the positive correlation (positive scene with positive template) with the negative correlation (negative scene with negative template). See Figure 12 for a flowchart of the correlation implementation.

PROCESS EVALUATION

The function resulting from the correlation process next needs to be evaluated. The correlation function can be evaluated by viewing it, or by a numerical analysis of its peaks.

PLTTRNS, by Schafer, enables the user to view 3-D, contour, and row plots on the Tektronix 4010 graphics terminal. The capabilities of PLTTRNS are enhanced by ITOC, which, among other tasks, converts integer files to complex files usable by PLTTRNS. See the source code listing in the appendix for further information on the use of ITOC.

The plots generated by ITOC and PLTTRNS give a rough idea of the success of the global search. PEAK gives
FIGURE 12: FLOWCHART OF CORRELATION IMPLEMENTATION.
quantitative information on up to ten supra-threshold correlation plane peaks: location, width, length, and value. More peak information is generated internally by PEAK if further peak discrimination is desired. As the correlation functions generally are not monotonically non-decreasing functions toward the absolute peak value, the selection of a threshold value is not a straight-forward task.

DISTANCE uses the peak locations found by PEAK to calculate a set of distance factors between the template and the scene window. The distance factors are based on the L1 and L2 norms, and take into account the scene and template energies, the template average pixel value, and the window size. The factors can be computed for the 256x256 pixel original images, or the 128x128 pixel reduced images used in the correlation process. The factors are scaled into a 0-100 range. See Figure 13 for a flowchart of the evaluation process.

SUPPORT SUBROUTINES

Several subroutines are common to many programs; they will be briefly described below.

TIMER is used to measure the execution time of a program in hours, minutes, and seconds. The first call to TIMER starts the "stopwatch," and the second call stops it. The run time is printed on the console.
EVALUATION

from correlation process
NPROD.IN

quantitative

PEAK

candidate windows

PSCEME.VD or RSCENE.VD

PTEMP.VD or RTEMP.VD

DISTANCE

distance scores

qualitative

ITOC

[CTOV]

distance scores plots

[PICTURE, VIDEO7]

FIGURE 13: FLOWCHART OF EVALUATION PROCESS.
IOF is used to allow switches and filenames to be read in the execution command line. IOF is a powerful device when macro (or automatic) programs are run.

REPACK is the routine to convert unpacked arrays into packed arrays before they are written to disk. UNPACK converts packed arrays read from disk into unpacked to be processed.

XRDBLK is used to read a packed block (256 words) from disk, unpack it, and return the unpacked array (1024 words) to the calling program.

In the next chapter, observations will be made on some results of the correlation process.
VI. OBSERVATIONS

In this chapter, the behavior of the detection process is briefly discussed, and demonstrated with several appropriate examples. In particular, the problems of dealing with the clutter energy, variable target illumination, window positioning errors, and window classification are considered. (See the appendix for more test results.)

CLUTTER ENERGY

The results of some early testing led to several modifications of the correlation process. Consider the cross-correlation functions (CCF) between a template and a scene with a target in it, shown in Figure 14, and the CCF between a template and a scene without a target, shown in Figure 15. The template and both scenes have energies of unity (globally normalized). In neither case are there clearly defined peaks in the CCF to suggest possible target locations. The maximum function values in both cases corresponded to windows of light background only, rather than to "interesting" objects such as trees or tanks. This behavior is attributed to the dominance of the high-energy background. Relatively low energy objects will not correlate well with the template, regardless of their forms. This failure is a classic weakness of the non-normalized CCF.
FIGURE 14: POSITIVE CCF FOR TANK SCENE TEST. VD AND TEMPLATE H3.
GLOBAL NORMALIZATION USED.

FIGURE 15: POSITIVE CCF FOR SCENE 13 AND TEMPLATE H3.
To take advantage of the a priori knowledge that the object of interest will have low-energy content, the CCF's were computed using the negatives of the scene and template. These are shown in Figures 16 and 17. The peaks of the functions are much more distinct than those of the positive CCF's. Notice especially the center peak of Figure 16, which corresponds to a tank in the scene.

The negative CCF's appeared to be an improvement, but two problems still remained. First, the digitizer noise of columns 1-8 resulted in the line of peaks seen at the right hand side of the negative CCF's. This line can clearly be seen in the top 20% contour plot in Figure 18. The second problem was that the false peaks in Figure 16 had values as large as the true peak (the true peak is the single peak corresponding to the target). One method to overcome these problems is to investigate each candidate peak by recomputing the CCF for a smaller sized scene window corresponding to that peak. Each new CCF is then investigated for the occurrence of distinct peaks, and the process continues until one window in the scene is chosen as being most likely to contain a target. Then a decision is made as to the content of the window.

The process of iteratively "windowing-in" on a target can be successful because the target-to-clutter area ratio
MAXIMUM = .319189
FULL WINDOW NEGATIVE CROSS-CORRELATION

FIGURE 16: NEGATIVE CCF FOR TANK SCENE TEST, VD AND TEMPLATE H3.
TARGET-TO-CLUTTER AREA RATIO IS 0.12.

NPROD - THRESHOLD = 0%

FIGURE 17: NEGATIVE CCF FOR SCENE 13 AND TEMPLATE H3.
increases with each successive CCF computed. Consider Figure 16, in which the target-to-clutter area ratio was 0.12. Notice (in Figure 19) the dramatic improvement obtained by increasing the target-to-clutter area ratio to 0.40.

An alternate method of choosing candidate windows is desired, as the windowing process can become computationally prohibitive for large scene areas. For this reason, the scene was locally normalized by grid blocks, in hopes of obtaining a distinct true peak with just one CCF calculation. In all test cases a 4x6 normalization grid was used. Rows 121-128 and columns 1-8 of the reduced scenes were set to zero prior to normalization to remove the possibility of digitizer noise dominating the correlation function. The effect of these two changes can be seen by comparing Figure 20 with Figure 17.
The distance factors corresponding to the peaks of the CCF of Figure 20 are given in Table III. Notice the behavior
of the score factor as the search window is shifted. The scene tested did not have a target present in this case.

TARGET ILLUMINATION

The grid method of normalization was next tested on scene PTANKH3. This is the same scene that was used to create template H3 (PTEMPH3). The scene was re-digitized to test the effect of the noise added in the digitization process. See Figure 21 for the digitized version of PTANKH3.

![Digitized Tank Scene H3](image)

**FIGURE 21: DIGITIZED TANK SCENE H3.**

The CCF between PTANKH3 and PTEMPH3 is shown in Figure 22. Unfortunately, there are no distinct peaks present to
### TABLE III. PEAK EVALUATION OF NPROD13, AND CORRESPONDING DISTANCES

<table>
<thead>
<tr>
<th>PEAK #</th>
<th>PEAK</th>
<th>H1</th>
<th>W1</th>
<th>L1</th>
<th>NPROD13</th>
<th>DISTANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>2.00</td>
<td>2.00</td>
<td>1.00</td>
<td>5.00</td>
</tr>
<tr>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>6.00</td>
</tr>
<tr>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>7.00</td>
</tr>
</tbody>
</table>

### RECOGNITION RESULTS

<table>
<thead>
<tr>
<th>TEMPLATE WITHIN</th>
<th>LENGTH</th>
<th>45 ROWS</th>
<th>TOP ROW</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>CENTERED VD</td>
<td>75</td>
<td>24</td>
<td>LEFTCOL</td>
<td>97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PEAK</th>
<th>PEAK</th>
<th>COLUMN</th>
<th>WIDTH</th>
<th>LENGTH</th>
<th>NPROD13</th>
<th>DISTANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>2.00</td>
<td>2.00</td>
<td>1.00</td>
<td>5.00</td>
</tr>
<tr>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>6.00</td>
</tr>
<tr>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>7.00</td>
</tr>
</tbody>
</table>

**COMMENT:** WINNER FROM PEAK EVALUATION IS NPROD13 (THRESHOLD=0.90X)

**COMMENT:** PEAK VALUE=1-100%, PEAK VALUE=2-50%, PEAK VALUE=3-10%
suggest further investigation. Apparently the grid rectangle size was not small enough to sufficiently normalize the high energy non-target areas. For a 4x6 grid, the grid rectangle size is 30x20 pixels, for an area of 600 pixels. A finer grid may produce a more meaningful CCF for the case when the target average energy per pixel is less than the clutter average energy per pixel. The effect of varying the grid size was not studied in this thesis.

The CCF of the negative of PTANKH3 (NTANKH3) and NTEMPH3 did exhibit a sharp peak, with a maximum value at (129,129). This coordinate pair corresponds to a registration shift error of only one pixel in each shift direction. See Figures 23 and 24 for the 3-D and contour plots. The peaks found, with corresponding distances, are listed in Table IV. The distance computed for Peak #1 from the reduced scene
comparison indicated the best match for a shift of (128,128), as was expected. The original 256x256 scenes were compared using (128,128) as the peak location, and a distance score of 35 was computed.

**FIGURE 23:** CCF OF W remake and WtempH3 (WprodN3).

**FIGURE 24:** Top 30% contour plot of WprodH3.
### TABLE IV. PEAK EVALUATION OF NPROH3, WITH CORRESPONDING DISTANCES

<table>
<thead>
<tr>
<th>PLAN</th>
<th>RAW</th>
<th>PEAK</th>
<th>COLUMN</th>
<th>WIDTH</th>
<th>LENGTH</th>
<th>NORMALIZED PVALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>20</td>
<td>20</td>
<td>694</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>20</td>
<td>20</td>
<td>512</td>
</tr>
</tbody>
</table>

**Template Window**  
LENGTH: 23 ROWS
TOP ROW: 173
WIDTH: 47 COLUMNS
LEFT COLUMN: 177

**Reduced Scene File ---> RPTAM03. VD**

<table>
<thead>
<tr>
<th>CORRELATION PLAN</th>
<th>WINDOW CENTER ROW Column</th>
<th>TOP ROW Column</th>
<th>LEFT COLUMN</th>
<th>CORRELATE FACTOR</th>
<th>L2 FACTOR</th>
<th>L1 FACTOR</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>129, 129</td>
<td>43</td>
<td>43</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td>34</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>35</td>
<td>35</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
<td>36</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
<td>37</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Comment:** Windows from Peak Evaluation of NPROH3

<table>
<thead>
<tr>
<th>CORRELATION PLAN</th>
<th>WINDOW CENTER ROW Column</th>
<th>TOP ROW Column</th>
<th>LEFT COLUMN</th>
<th>CORRELATE FACTOR</th>
<th>L2 FACTOR</th>
<th>L1 FACTOR</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38, 38</td>
<td>43</td>
<td>43</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>39</td>
<td>39</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>40</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
<td>41</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>42</td>
<td>42</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Comment:** Windows correlated from distance results with peak @ 129, 129

53
One of the reasons for this low score can be attributed to the method of normalization used prior to the distance calculation. As the template window and the target have been given to be the same size, the search window will contain only target information when properly placed. The clutter energy will be eliminated, so the only normalization thought to be needed was search window normalization (in other words, a 1x1 grid normalization). This type of normalization will not take into account energy changes over the target due to illumination. Recall that the brightness function is a product of the reflectance and illumination functions. It is an implicit assumption that the target reflectance function is expected to be about the same as that of the template. To take into account the variability of the illumination, both the scene and the template should be normalized by a method similar to that used in computing the CCF. Grid normalization to improve the confidence of the distance factors was not implemented due to time constraints. However, a simple demonstration of the advantage of using search window grid normalization is given in Table V. Note the increase in the best distance score factor from 35 to 79.

POSITIONING ERRORS

The correlation process was next tested using NTANKD2 as the input scene, and NTEMPH3 as the template. The results of the peak evaluation of NPRODD2 are given in Table VI.
scene D2 and the window choices of PEAK are shown in Figures 25 and 26. The distances corresponding to the windows chosen by PEAK are given in Table VII. Notice that the first window choice clearly corresponds to the target, but that the distance factors favor the second window choice. As expected, the distance factors are very sensitive to window positioning errors, since they are based only upon a pixel-by-pixel comparison.

To achieve smaller window positioning errors, a second CCF can be computed between the template and an intermediate size window. The window size would be larger than the template, but much smaller than the original scene area. An improvement of this type would be relatively computationally inexpensive, as the FFT based correlation time is proportional to \( N \times N \log N \), where \( N \) is the window width.
FIGURE 25: DIGITIZED TANK SCENE D2.

FIGURE 26: FIRST TWO WINDOW CHOICES FROM PEAK EVALUATION OF NPRODD2.
### TABLE VI. PEAK EVALUATION OF NPROD2

<table>
<thead>
<tr>
<th>PEAK</th>
<th>XMAX</th>
<th>ROW</th>
<th>COLUMN</th>
<th>WIDTH</th>
<th>LENGTH</th>
<th>NORMALIZED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>129</td>
<td>121</td>
<td>33</td>
<td>17</td>
<td>.758</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td>99</td>
<td>64</td>
<td>6</td>
<td>3</td>
<td>.546</td>
</tr>
</tbody>
</table>

---

### TABLE VII. DISTANCES CORRESPONDING TO WINDOWS OF FIGURE 26

#### RECOGNITION RESULTS

*REDUCED***

**TEMPLATE WINDOW:** (RPTENPH2.VD)  
LENGTH = 23 ROWS  
WIDTH = 47 COLUMNS  
TOP ROM = 173  
LEFT COL = 177

*RREDUCED* SCENE FILE --> RPTANK2.VD

<table>
<thead>
<tr>
<th>PEAK (ROM, COLUMN)</th>
<th>CENTER (ROM, COLUMN)</th>
<th>TOP (ROM, COLUMN)</th>
<th>LEFT (COLUMN)</th>
<th>CORRELATE FACTOR</th>
<th>L2 FACTOR</th>
<th>L1 FACTOR</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 129, 121</td>
<td>36, 78</td>
<td>47</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>36, 79</td>
<td>47</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>36, 80</td>
<td>47</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>99, 78</td>
<td>48</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>99, 79</td>
<td>48</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>99, 80</td>
<td>48</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>60, 78</td>
<td>49</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>60, 79</td>
<td>49</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>60, 80</td>
<td>49</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2: 99, 84</td>
<td>84, 115</td>
<td>73</td>
<td>92</td>
<td>21</td>
<td>11</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>84, 116</td>
<td>73</td>
<td>93</td>
<td>19</td>
<td>10</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>84, 117</td>
<td>73</td>
<td>94</td>
<td>17</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>85, 115</td>
<td>74</td>
<td>92</td>
<td>22</td>
<td>12</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>85, 116</td>
<td>74</td>
<td>93</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>85, 117</td>
<td>74</td>
<td>94</td>
<td>18</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>86, 115</td>
<td>75</td>
<td>92</td>
<td>22</td>
<td>12</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>86, 116</td>
<td>75</td>
<td>93</td>
<td>20</td>
<td>11</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>86, 117</td>
<td>75</td>
<td>94</td>
<td>18</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

**COMMENT:** PEAK 1 CORRESPONDED TO THE TARGET.  
**COMMENT:** PEAK 2 WAS 72% OF PEAK 1.
Thus, the correlation computation time quickly diminishes as \( N \) is decreased.

**CLASSIFICATION ERRORS**

The ability of the distance factors to classify windows was tested using PTANKD2, and the standard template, PTEMPH3. A window was selected in PTANKD2 to correspond to the tank only (the ideal choice of PEAK evaluating NPRODD2).

None of the distance factors of the nine windows investigated by DISTANCE were greater than zero, a somewhat disappointing result. There are several possible explanations for this result. The first is that scene D2 and template H3 were digitized in separate sessions. As a result, the size and orientation of the tanks will slightly differ. The second explanation is that the normalization approach used in the distance calculation admittedly does not account for illumination changes over the target, as mentioned previously. Finally, the 16 level quantization cannot accurately represent the continuous image unless the brightness function histogram is reasonably "spread out."

The third explanation requires further discussion. Consider the histograms of template H3, and of the tank of scene D2 (Figures 27 and 28). The histogram of template H3 is almost evenly distributed from levels 0 to 9. On the other hand, the histogram of the tank of scene D2 shows 63%
of the pixels to be in levels 2 and 3. This "distortion" from the ideal (template) histogram will result in some loss of detail in the digitized image. Recall that one of the assumptions made is that the digitized images must be accurate representations of the continuous scene, or dependable detection cannot be expected.

The histogram of RTEMPD2 was modified by TONER so that any detail in the scene would be enhanced. The pixel mapping is shown in Table VIII, and the resulting histogram is given in Figure 29. The enhanced version of TANKD2 is compared with template H3 in Figure 30. The effect of the difference in illumination can clearly be seen by comparing the top
portions of the tanks, and also the bottom portions. Unless an elegant method of normalization is used, any point-by-point based distance measure may give misleading results.

There are several methods of computing distances that take into account the spatial relationships of the pixels in an image. Although none were implemented in this study, they
**TABLE VIII. PIXEL MAPPING FOR TONED SCENE D2**

**RESULTS OF TONER**

Input file —> ATEMP02.VD  
Output file —> TONEDD2.VD

<table>
<thead>
<tr>
<th>OLD PIXEL</th>
<th>NEW PIXEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

**FIGURE 29: HISTOGRAM OF TONEDD2.VD**
are suggested as logical improvements to the detection process. One method to be tested is to compare in some way the filtered Fourier components of the normalized scene window with those of the normalized template window. Another possibility to explore is to compute a statistical correlation measure of the form

$$Ng(m,n) = \frac{\sum \sum f(s(x,y))f(t(x-m,y-n))}{\sqrt{\sum \sum \{f(s(x,y))\}^2 \sum \sum \{f(t(x-m,y-n))\}^2}}$$

(49)

where $f(\cdot)$ indicates some 3x3 mask operation of the array, most likely an edge enhancer. Linear edge enhancers include the Laplacian mask, while non-linear enhancers include the Kirsch and Roberts operators [5: 482-491]. A 3x3 mask operator can easily be implemented by modifying the subroutine TEST3 of the program REMOVE.
VII. CONCLUSION

SUMMARY

In this thesis, a method for two-dimensional pattern recognition was developed and tested. The method included a global search scheme for candidate targets, based on high speed cross-correlation between a normalized scene and template. A target classification measure dependent on the normalized L1 and Euclidean distances was also presented.

Several computer programs were written to carry out the process, from image input to target classification. Especially significant was the program PEAK, which performs a search of two-dimensional cross-correlation functions for isolated supra-threshold peaks. Another program, DISTANCE, computes a classification, or similarity score between template and scene windows corresponding to the correlation peaks. Simple image processing programs were also written and described.

CONCLUSIONS

1. The global search for candidate targets by cross-correlation template matching can be counted on to find candidate targets to only a limited degree. The search scheme presented in this study is best used to determine the approximate location of amorphous "blobs."
2. The reliability of cross-correlation in finding targets increases as the target-to-clutter area ratio increases.
This property can allow the target to be found by an iterative process of correlations using smaller and smaller window sizes.

3. Similarity measures based on pixel-by-pixel comparisons are sensitive to slight mis-registration errors, and to minor degradations in the brightness function due to illumination and digitization noise.

4. The grid normalization method of chapter four for use in the correlation process is not a suitable approximation of search window normalization. The method has difficulties when a grid rectangle covers an area which contains both high and low energy objects. However, it works very well for normalizing scenes and templates in the classification process, after a search window has been chosen.

RECOMMENDATIONS

1. An algorithm to automatically evaluate the information generated by the program PEAK needs to be developed. Specifically, a simple rule to determine the selection of a threshold value (or possibly several values for a given CCF) needs to be determined. Also to be determined is a method to recognize a false peak (one not corresponding to a target), given its width, length, and percentage of maximum peak value. Perhaps the cross-section of the peak could be compared with the cross-section of the autocorrelation function. Elimination of false peaks from further consideration would result in computational savings in the
classification process.

2. A more elegant normalization scheme is needed for use in the global search process. One possibility would be to compute a finer array of normalization constants that take into account the energies of the surrounding grid rectangles (see Figure 31). The normalization coefficients computed would essentially be the result of overlapping the grid rectangles.

3. An intermediate correlation using a smaller window area should be computed, using the peak information from the global search CCF to determine the window size and center. This intermediate correlation will allow for a precise positioning of the window used in the computation of the similarity measure.

4. The use of grid normalization of scene and template windows for the classification process needs to be further developed and tested. Preliminary results indicate that grid normalization can be used successfully to account for differences in energy between the template and scene tanks, as long as most of the clutter energy is eliminated (which is the case when the search window is accurately positioned).

5. The choice of features to be compared needs to be studied. Instead of comparing the pixel values of the scene candidate window directly with those of template window, the comparison of the high or low-frequency Fourier components, for example, may lead to more promising results. It might also be interesting to compute the distances between scenes
FIGURE 31: SUGGESTED NORMALIZATION SCHEME

Consider the case where the energies of 9 local rectangular "whole" regions A-I are computed.

\[
\begin{array}{ccc}
A & B & C \\
D & E & F \\
G & H & I \\
\end{array}
\]

From these 9 energy terms, we want to compute 81 normalization coefficients. The coefficients for any "sub"-region can be approximated by a weighted average of the whole region coefficients corresponding to the center sub-region and its eight nearest neighbors, where the weights are determined by the distance from the center sub-region.

\[
N(A') = \sqrt{K_1xA + K_2x[(A + A + B + D) + (A + B + E + D)/\sqrt{2}]}
\]

This normalization method may avoid some of the severe discontinuities sometimes created by whole region normalization.
and templates that have been mask processed (for example edge enhanced), as mentioned in chapter six.
BIBLIOGRAPHY


APPENDIX A:

USE OF MACRO FILES

Under the RDOS operating system, macro (or indirect files) can be run to execute a series of CLI commands. Thus, a process consisting of a series of program can be run automatically as long as none of the programs require interactive user input. In order that file names and program options can be specified in the execution line, all of the programs in the correlation process use the COMARG call to read the command line argument string. This appendix describes the use of several macro programs.

**NTEMPCP.MC**

Macro NTEMPCP.MC creates NTEMP.CP, the Fourier transform of the reduced template, RNTEMP.VD. RNTEMP.VD is the negative of the 256x256 PTEMP.VD, reduced to the lower right quadrant by REDUCE and NMOVE. Links to NORMALIZE and DIRECT must exist before NTEMPCP.MC is executed. The program lines are as follows:

```
NORMALIZE/L RNTEMP.VD NTEMP.CP
DELETE/V RNTEMP.VD
DIRECT NTEMP.CP/I 256/N
```
Execution of NPRODIN.MC results in the CCF between the template and SCENE.VD to be computed and stored in NPROD.IN. Files HOLD1.CP and HOLD2.CP should exist as 1024-block contiguous files; they will be created if they do not exist. The use of contiguous (as opposed random) files decreases run time by a factor of 3. The following files must be linked to before NPRODIN.MC can be run (the links in CROMER.DR are given as an example):

CROMER
:26:26

HOLD1.CP STROUPE:HOLD1.CP
HOLD2.CP STROUPE:HOLD2.CP
NTEMP.CP STROUPE:NTEMP.CP
SCENE.VD STROUPE:SCENE.VD
NEGATE.SV STROUPE:NEGATE.SV
REDUCE.SV STROUPE:REDUCE.SV
NORMALIZE.SV STROUPE:NORMALIZE.SV
DIRECT.SV DP4F:DIRECT.SV
CMULTIPLY.SV STROUPE:CMULTIPLY.SV
INVERSE.SV DP4F:INVERSE.SV
CTOI.SV STROUPE:CTOI.SV

Run time will be less than 10 minutes. The scene of interest should be renamed SCENE.VD before executing; NPROD.IN should be renamed after the macro program execution. See COMPUTE.MC for further clarification on the use of NPRODIN.MC. One version of NPRODIN.MC is as follows:

RENAME HOLD1.CP NSCENE.CP
RENAME HOLD2.CP NPROD.CP
NEGATE SCENE.VD NSCENE.VD
REDUCE NSCENE.VD RNSCENE.VD
DELETE/V NSCENE.VD
NORMALIZE/UNSCENE.VD NSCENE.CP
DELETE/UNSCENE.VD
DIRECT NSCENE.CP/I 256/N
CMULTIPLY NTEMP.CP NSCENE.CP NPROD.CP
RENAME NSCENE.CP HOLD1.CP
INVERSE NPROD.CP/I 256/N
CTOI NPROD.CP NPROD.IN
RENAME NPROD.CP HOLD2.CP

COMPUTE.MC

COMPUTE.MC is used to automatically, without user interaction, compute the CCF's corresponding to six different scenes and one template. The total run time is less than one hour. The macro controls the re-naming (or re-linking in this case) of the dummy files SCENE.VD and NPROD.IN. The CCF's created are moved to other disks to allow room to run the rest of the correlations. The macro program requires 256K bytes of free disk (assuming NPROD.IN exists). If for any reason NPROD.IN cannot be moved, it is simply overwritten. The program may be aborted at any time the user desires; results up to that time will be saved. The program is given as follows:

LINK SCENE.VD CROMER:PTANKQ4.VD
NPRODIN
RENAME NPROD.IN NPRODG4.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKD4.VD
NPRODIN
MOVE/V/D/R DP5F NPROD.IN/S NPRODD4.IN
LINK NPRODD4.IN DP5F:NPRODD4.IN
UNLINK SCENE.VD
LINK SCENE.VD CROMER:PTANKC3.VD
NPRODIN
MOVE/V/D/R DP5F NPROD.IN/S NPRODC3.IN
The scene files correlated with the template are as follows:

**CROMER**

- PTANKB2.VD 32768 PC
- PTANKC3.VD 32768 PC
- PTANKD2.VD 32768 PC
- PTANKD4.VD 32768 PC
- PTANKE2.VD 32768 PC
- PTANKG4.VD 32768 PC

The files and links created by this version of COMPUTE.MC are as follows:

**CROMER**

- NPRODB2.IN  DP5F:NPRODB2.IN
- NPRODC3.IN  DP5F:NPRODC3.IN
- NPRODD2.IN  DP5:NPRODD2.IN
- NPRODD4.IN  DP5F:NPRODD4.IN
- NPRODE2.IN  DP5:NPRODE2.IN
- NPRODG4.IN  131072 C
# APPENDIX B: SUMMARY OF PROGRAM USAGE

<table>
<thead>
<tr>
<th>INPUT FILE TYPE</th>
<th>EXECUTION LINE FORMAT</th>
<th>OUTPUT FILE TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[.VD]</td>
<td>VIDEO7</td>
<td>[.VD]</td>
</tr>
<tr>
<td>.VD</td>
<td>QUICKAVG3</td>
<td>AVERAGE7.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>REMOVE infile out1 out2</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>EVIDHIST</td>
<td></td>
</tr>
<tr>
<td>.VD</td>
<td>TONER</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>REMOVE</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>NEGATE[/F] infile outfile</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>REDUCE infile outfile</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>NORMALIZE[/U or /L] infile outfile</td>
<td>.CP</td>
</tr>
<tr>
<td>.CP</td>
<td>DIRECT infile/I 256/N [outfile/O]</td>
<td>.CP</td>
</tr>
<tr>
<td>.CP</td>
<td>CMULTIPLY infile1 infile2 outfile</td>
<td>.CP</td>
</tr>
<tr>
<td>.CP</td>
<td>INVERSE infile/I 256/N [outfile/O]</td>
<td>.CP</td>
</tr>
<tr>
<td>.CP</td>
<td>CTOI infile outfile</td>
<td>.IN</td>
</tr>
<tr>
<td>.IN</td>
<td>IMULTIPLY infile1 infile2 outfile</td>
<td>.IN</td>
</tr>
<tr>
<td>.IN</td>
<td>ITOC/[A,N,E,H or 0] infile[/C][/T] outfile[/M]</td>
<td>.CP</td>
</tr>
<tr>
<td>.CP</td>
<td>PLTRNS infile/I 256/N</td>
<td></td>
</tr>
<tr>
<td>.IN</td>
<td>PEAK</td>
<td></td>
</tr>
<tr>
<td>.VD</td>
<td>DISTANCE</td>
<td></td>
</tr>
<tr>
<td>.CP</td>
<td>CTOV</td>
<td>.VD</td>
</tr>
<tr>
<td>.VD</td>
<td>PICTURE</td>
<td></td>
</tr>
</tbody>
</table>

where

- **.VD** — 32K bytes packed video
- **.IN** — 128K bytes integer
- **.CP** — 512K bytes complex
- [ ] — optional input or output

73
This appendix contains the following programs (subroutines given in parenthesis):

1. VIDEO7 (CHAN7, VABORT, ERCHK)
2. QUICKAVE7
3. PICTURE (OUT3X3, OUT4X3)
C***** NOVA VIDEO7 INPUT/OUTPUT ROUTINE VIA CROMEMCO COMPUTER *****

WRITTEN BY Lt. Jim Cromer 12 Aug 1982

Fortran IV

This program allows the user to display a video file repeatedly any number of times. It also allows the user to input or output video files named A0, A1, A2, A3, A4, A5, etc. automatically. These files may then be averaged by QUICKAVE7.

Execution Line Format:

VIDEO7

Load Line Format:

RLDR VIDEO7 CHAN7 ERCHK VABORT CHANNEL

DCHRX DCHTX SANDS CANDR FORT.LB

C**********************************************
DIMENSION IPAR(2), IHOLD(7)
INTEGER FILE(7)
IPAR(2) = 0

C******** USER PARAMETER INPUT **********************

TYPE"NOTICE: CHOPS must be running! <12>"
ACCEPT"Input or output (IN-0/OUT-1)?", IDIR
IF(IDIR.NE.0.AND.IDIR.NE.1)GO TO 8 ; Error checking

ACCEPT"Enter time (SEC.): ", ITIME ; Monitor display time
IF(IDIR.EQ.0)GO TO 4
ACCEPT"Type 1 to output A0-An: ", IDIR
IF(IDIR.EQ.1)GO TO 4
ACCEPT"What is the name of the data file (13 Char max)? ", FILE
READ(11,8)FILE(I)
ACCEPT"Enter # times to be displayed: ", II
IF(FILE(I).NE.10752)GO TO 42 ; An "*" means to DO 2 I=1,7 ; run with the last

FILE(I)=IHOLD(I) ; file used
GO TO 42

C******** INPUT OR OUTPUT FILES A0-A6 ********************

IPAR(I)=ITIME
CALL CHAN7(IDIR,IPAR)
GO TO 6

C******** DISPLAY VIDEO FILE OF USER'S CHOICE **************

DO 43 KK=1,II
TYPE"CHECK MONITOR -- ", KK
CALL VABORT(NISK, IM, IPQNT, IDCNT)

75
IDIR=1
IPAR(1)=ITIME
IERR=0
CALL CHANNEL(MISK, IDIR, IM, IPQNT, IDQNT, FILE, 64, 0, IPAR, IERR, ISYS)
CALL ERRCHK(IERR, IDIR, IDQNT, IPQNT, ISYS) ; identify errors

C

C******** USER INPUT ***********************************************

C

6 TYPE"<12>"
DO 7 I=1,7 ; Save filename
7 IHOLD(I)=FILE(I) ; for next loop
ACCEPT"What next (INPUT-0,OUTPUT-1,STOP-2)? ",IDIR
IF(IDIR.EQ.0) GO TO 1
IF(IDIR.EQ.1) GO TO 1
IF(IDIR.EQ.2) STOP "Type VIDEO7 to rerun."
8 FORMAT(S13) ; Filename input format
TYPE"<7>"Input error; try again."
GO TO 6
END

C

C********** Program VIDEO7 ***************************************

76
SUBROUTINE CHAN7(IB, IPAR) ;by Lt Jim Cromer

C******************************************************************************

C Subroutine CHAN7 will digitize a picture seven times, and
C will store the video data in files A0-A6 (i.e. in a format
C usable by QUICKAVE7). It will also display the digitized
C pictures consecutively on the video monitor.
C (Called by VIDE07)

PARAMETERS PASSED:

IB=0 ----> input pictures from camera
IB=1 ----> output pictures to the video monitor
IPAR(1) ---> display time in seconds
IPAR(2) ---> unused

******************************************************************************

INTEGER FILE(7), IPAR(2)
DO 100 I=1, 7
100 FILE(I)=0
IERR=0
IF(IB.EQ.1) GO TO 300
DO 200 I=16688, 16694 ;if digitizing, delete A0-A6
   FILE(I)=I
200 CALL DELETE(FILE)
IF(IB.EQ.1) GO TO 400
300 TYPE "Enter the number of pictures to be displayed."
   TYPE "They must be named A0-A(N-1) to be displayed."
   ACCEPT ENTER "", INUM
   IF(INUM.GT.10) TYPE "Sorry. The maximum number of pictures
   $ is 10."
   IF(INUM.GT.10) INUM=10
400 DO 500 I=1, INUM ;if I=1, then FILE="A0"
   ;if I=2, then FILE="A1"
   FILE(I)=16687+I ;and so on
   CALL VABORT (IA, IC, ID, IE)
   WRITE (10, 600) FILE(I)
      CALL CHANNEL (IA, IB, IC, ID, IE, FILE, 64, 0, IPAR, IERR, ISYS)
   CALL ERCHK (IERR, IE, ID, ISYS)
500 FORMAT (" Check monitor — Picture being displayed —> ", SL3)
RETURN
END

C******************************************************************************

77
SUBROUTINE VABORT(NISK, IM, IPCNT, IDCNT)

C---------------------------------------------------------------------------
C
C This subroutine sends an abort call to CHANNEL,
C and resets the calling parameters for the next
C VIDEO7 call.
C
C************************************************************************

J=0 ;dummy variable
K=3 ;mode 3 --> Abort
IDCNT=4
IPCNT=1 ;parameter count
NISK=3
IM=2 ;mode 2 --> Video
RETURN

END
C
C*********** Subroutine VABORT **********************************************

78
SUBROUTINE ERCHK(IERR, IDIR, IDCNT, IPQNT, IYS)
C**************************************************************************
C
This subroutine checks for errors made during an attempted
call to CHANNEL, and prints them to the screen.
(Called by VIDE07).

C**************************************************************************
INTEGER CRERR, PDCONT
LOGICAL BTST
IF (IERR.EQ.0 OR (IERR.EQ.13323 .AND. IDIR.EQ.0)) GO TO 6    ;ok
IF (IERR.EQ.13323 .OR. IERR.EQ.-24832 .OR. IERR.EQ.-24064 .OR. IERR
* .EQ.-22528) GO TO 5    ;Specific error messages will be given
IF (BTST(IERR,15)) GO TO 9    ;Abnormal return
GO TO 10    ;Error without abort
5 IF (IERR.EQ.-24832) TYPE"<7>ABORT—FILE DOES NOT EXIST"    ;Specific error messages will be given
IF (IERR.EQ.-24064) TYPE"<7>ABORT—FILE DOES NOT
* CONTAIN VIDEO"
IF (IERR.EQ.-22528) TYPE"<7>ABORT—FILE CANNOT BE CREATED"
6 TYPE"<7>Channel cleared"    ;Specific error messages will be given
GO TO 20
9 TYPE"<7> ABORT INITIATED!!!<12>"    ;Abnormal return
10 CRERR=15 AND,lERR    ;Error
PDCONT=LSHT(240, AND, IERR,-4)    ;two error
NOVERR=LSHT(-256, AND, IERR,-8)    ;codes in the
TYPE" CROMEMCO ERROR RETURNED: ",CRERR    ;variable
TYPE" PARAMETER/DATA COUNT RETURNED: ",PDCONT
CALL BCLR(NOVERR,7)
TYPE" NOVA ERROR RETURNED: ",NOVERR    ;messages are
TYPE" ERROR CODE RETURNED: ",IERR    ;printed for
TYPE" DATA COUNT: ",IDCNT    ;user information
TYPE" PARAMETER COUNT: ",IPQNT    ;and correction
PAUSE
20 RETURN
END
C
************************************************************************** Subroutine ERCHK**************************************************************************
Program QUICKAVE7  Written by Lt. Jim Cromer
Fortran 5 (by DATA GENERAL)

This program will average the packed video files
"A0" - "A6" pixel by pixel, and output the averaged picture
to the packed video array "AVERAGE.VD". [A packed video
array contains a 256x256 4-bit/pixel picture in a 64 block file
on disk, where 1 block=256 16-bit words (i.e. 1 block=4 video
rows). Each 16-bit word holds 4 pixels. Video files are
stored on disk in packed form to conserve disk space. File
size is 32K bytes.] Total run time is less than one
minute clock time.

Execution Line Format:  (run SLOWAVE7 on the NOVA)
QUICKAVE7

Loader Command Line Format:
RLDR QUICKAVE7 TIMER UNPACK REPACK 20/C @LIB@

The 20/C opens up enough channels to run the
program [by default, only 8 channels (designated and pre-
designated) are normally available]. Links to A0-A6 in
DPOF should be created.

C******** ARRAY MANAGEMENT ****************************

Arrays A0-A6 hold 2 blocks each (i.e. 8 packed video rows)
Arrays A0U-A6U hold 8 blocks each (i.e. 8 unpacked video rows)
Array AVE is used for the unpacked averaged picture (8 rows)
Array AVEP holds the packed averaged picture (8 rows)

[If array sizes are modified, note that arrays A*U must be dimen-
sioned to be 4 times larger than arrays A*.

The EQUIVALENCE statement is used to reduce memory
requirements. Once an array is unpacked, the packed array is
no longer used. Therefore unneeded packed arrays can be used
to hold unpacked arrays (but not at the same time).

INTEGER FILE(2)
INTEGER A0(512), A1(512), A2(512), A3(512), A4(512), A5(512),
$ A6(512), AVE(2048), A,B,C,D
INTEGER A0U(2048), A1U(2048), A2U(2048), A3U(2048), A4U(2048),
$ A5U(2048), A6U(2048), AVEP(512)
COMMON must be declare
;before EQUIVALENCE
;can be used (Fortran IV)
EQUIVALENCE (AVE, A0U), (A0, A1U), (A1, A2U), (A2, A3U), (A3, A4U), (A4, A5U)
$ (A5, A6U), (A6, AVEP)

TYPE* Program QUICKAVE7 now executing . . .<15><7>"

CALL TIMER(0) ;start timer

C******** I/O FILE MANAGEMENT ****************************
C
C Seven channels to "A0" thru "A6" must be OPENed for reading. An eighth channel to "AVERAGE7.VD" must be OPENed for writing (after "AVERAGE7.VD" is created).

FILE(2)=0
DO 998 I=0,6
   FILE(1)=16688+I
   IF(IER.EQ.1)WRITE(10,999)FILE(1),IER
C
999 CONTINUE
998 DELETE "AVERAGE7.VD"
C
CALL CFILW("AVERAGE7.VD",3,64,IER) ;create a contiguous file, if possible
IF(IER.NE.1)GO TO 997
C
CALL CFILW("AVERAGE7.VD",2,IER) ;create a random file of variable size
C
IF(IER.NE.1)TYPE " AVERAGE7.VD create error",IER STOP
C
CALL OPEN(7,"AVERAGE7.VD",3,IER) ;OPEN a channel to
C
IF(IER.NE.1)TYPE " OPEN 7 error ",IER ;"AVERAGE7.VD" for writing
C
DO 4 I=0,62,2 ;average 64 blocks, 2 at a time
C
C Read 2 blocks from each picture to be averaged (8 video rows)
C
C
CALL RD_BLK(0,I,A0,2,IER)
CALL RD_BLK(1,I,A1,2,IER)
CALL RD_BLK(2,I,A2,2,IER)
CALL RD_BLK(3,I,A3,2,IER)
CALL RD_BLK(4,I,A4,2,IER)
CALL RD_BLK(5,I,A5,2,IER)
CALL RD_BLK(6,I,A6,2,IER)
C
IF(IER.NE.1..OR..IER.NE.1..OR..KER.NE.1..R.IER.NE.1..OR.
MER.NE.1..OR..NER.NE.1..OR..I6ER.NE.1)GO TO 5
C
C Unpack the arrays
C
CALL UNPACK(512,A0,A0U)
CALL UNPACK(512,A1,A1U)
CALL UNPACK(512,A2,A2U)
CALL UNPACK(512,A3,A3U)
CALL UNPACK(512,A4,A4U)
CALL UNPACK(512,A5,A5U)

81
CALL UNPACK(512,A6,A6U)

Perform pixel by pixel averaging

DO 1 K=1,2048

AVE(K) = IFIX(FLOAT(A0U(K)+A1U(K)+A2U(K)+A3U(K)+A4U(K)+A5U(K))/7.0+0.5) ; round to nearest integer

1 CONTINUE

Pack array AVE into AVEP and write it to the disk

CALL REPACK(512,AVE,AVEP)

CALL WRBLK(7,I,AVEP,2,IER)

IF (IER .NE. 1) TYPE " WRBLK "$",I," error: ",IER

" Block "$",I," averaged. <15>

; can be deleted to decrease run time

CONTINUE

Print total run time message to the console CRT, and EXIT the program.

CALL TIMER(1) ; stop timer

GO TO 6

5 TYPE" <15>RDBLK A0 error: ",IER,"<15>RDBLK A1 error: ",I6ER,

"<15>RDBLK A2 error: ",I6ER,"<15>RDBLK A3 error: ",I6ER,

"<15>RDBLK A4 error: ",I6ER,"<15>RDBLK A5 error: ",I6ER,


6 CALL RESET

TYPE" <15> Program QUICKAVE7 finished." 

TYPE" The averaged file is named AVFRAGE7.VD<15><7><7>

STOP

END

***************************************************************************

Program QUICKAVE7 ***************************************************************************
This program will convert video pixels to lineprinter pixels, and will put the picture in a file or to the Printronix 300 lineprinter. This program prints either the complete 256 by 256 pixel picture, or a smaller picture that does not contain the noise created by the video digitizer (the last five blocks in the video file are noise). Odd numbered Video lines (rows) are represented by 3x3 pixels, while even numbered Video lines are represented by 4x3 pixels. Run time for a shortened picture of length 222 lines is about 1.5 minutes; the file size will be about 204Kbytes (or 102,000 words). For a 256x256 picture, run time should be 2 minutes, with 239Kbytes used on disk.

INTEGER ATTITLE(40)
DIMENSION ILARRAY(268), I2ARRAY(264), I3ARRAY(201), I4ARRAY(198)
EQUIVALENCE (ILARRAY, I2ARRAY, I3ARRAY, I4ARRAY)
DIMENSION ILP(4,67), IFILE(7), IRBC(64), ISAV(4)
LOGICAL SHORT, TITLE

Set up solid line, space, and line feed/plot-on characters

IL=177777K ; Solid line
NC=40100K  ; Space
LF=012K    ; Line feed
LFPO=2412K ; Line feed/plot on

Set up parameters for complete picture display.

SHORT=.FALSE. ; Short picture test
NL=66        ; Top and bottom border length
N2=256       ; Number of lines displayed
N3=1         ; Location of left border
N4=66        ; Location of right border
N5=67        ; Location of line feed
N6=1         ; Length of lines displayed

Open the video file for input

ACCEPT" What is the name of the input file? "
READ(11,17) IFILE(1) ; Read video file name
CALL OPEN(1,IFILE,1,IER) ; Open the video file
CALL CHECK(IER)
IF(IER.NE.1) TYPE" Input open error: ", IER

Ask for an output file, either the lineprinter or a disk file, and open the disk file if necessary.

ACCEPT" Do you want a disk file created (Y or N)? "
READ(11,19)N
*Read one ASCII character
IF(N.EQ.19968)GO TO 2
;File was not selected
IF(N.NE.22784)GO TO 20
;Input error
ACCEPT* A file was selected; what should its name be
* (13 char max)? *
READ(11,17)IFILE(1)
;Read output file name
CALL DFILW(IFILE,JER)
;Make sure that the
IF(JER.NE.1.AND.JER.NE.13)TYPE* Output delete error:*,JER
CALL CFILW(IFILE,2,KER)
;file does not exist
IF(KER.NE.1)TYPE* Output create error:*,KER
CALL OPEN(12,IFILE,3,LER)
;Open the output file
CALL CHECK(LER)
IF(LER.NE.1)TYPE* Output file open error:*,LER
GO TO 3
2 TYPE* The picture will only go to the lineprinter.*

Choose between complete picture and noiseless picture.

3 CONTINUE
ACCEPT*Do you want to title the picture (Y or N)? *
READ(11,19)N
IF(N.EQ.22784)TITLE=.TRUE.
DO 300 KT=1,40
   ATITLE(KT)=0
300 CONTINUE
IF(TITLE)TYPE*Enter title below (up to 80 characters).*
IF(TITLE)READ(11,3000)ATITLE(1)
3000 FORMAT(S80)
ACCEPT*Do you want a complete picture (Y or N)? *
4 READ(11,19)N
;Read one ASCII character
IF(N.EQ.19968)GO TO 22
;Response was "NO"
IF(N.NE.22784)GO TO 21
;Input error
TYPE* A complete 256 by 256 pixel picture was chosen.*

Put a border at the top of the picture.

5 DO 7 I=1,3
   IF(SHORT)WRITE BINARY(12)NC
   ;Space right
   IF(SHORT)WRITE BINARY(12)NC
   ;Space right again
   DO 6 J=1,NL
6 WRITE BINARY(12)IL
7 WRITE BINARY(12)LFFC
;Print a line
;Terminate the line

Each line of the picture will have a border on each end. A
DO-LOOP loops 233 (or 256 for whole frame) times around the
next three program parts

JTEST = -1
DO 13 JA=1,N2
JTEST = JTEST * -1
JJ = 4
IF(JTEST.GT.0)JJ=3 ;Odd iteration

Put a border down the left hand side

DO 8 K=1,JJ
    IF(SHORT)ILP(K,1)=NC
    IF(SHORT)ILP(K,2)=NC
    ILP(K,N3)=43500K
8

Put a border down the right hand side

DO 9 L=1,JJ
    ILP(L,N4)=40170K
    ILP(L,N5)=LPPC
9

Convert the video picture pixels to SLFT pixels

READ BINARY(1)IREC ;Read one video line
DO 12 N=N5,64
    INR=BYTE(IREC(N),2) ;Right two pixels
    IWR=BYTE(IREC(N),1) ;Left two pixels
    IF(JJ.GT.3.5)CALL OUT4X3(IWR,ISAV)
    IF(JJ.LT.3.5)CALL OUT4X3(INR,ISAV)
    N7=N+1
    IF(SHORT)N7=N
    DO 11 JB=1,JJ
    BYTE(ILP(JB,N7),1)=ISAV(JB) ;Move to high byte
    IF(JJ.LT.3.5)CALL OUT3X3(IWR,ISAV)
    IF(JJ.GT.3.5)CALL OUT4X3(INR,ISAV)
    DO 12 JC=1,JJ
    BYTE(ILP(JC,N7),2)=ISAV(JC) ;Store low byte
12
    L=0
    DO 130 JB=1,JJ
    DO 130 JD=1,N5
    L=L+1
    IARRAY(L)=ILP(JE,JD)
130
    CONTINUE
    IF(N5.EQ.66..AND.JJ.EQ.3)WRITE BINARY(12)IARRAY
    IF(N5.EQ.67..AND.JJ.EQ.3)WRITE BINARY(12)IARRAY
    IF(N5.EQ.66..AND.JJ.EQ.4)WRITE BINARY(12)IARRAY
    IF(N5.EQ.67..AND.JJ.EQ.4)WRITE BINARY(12)IARRAY
    CONTINUE

Put a border and title at the bottom of the picture

DO 15 JF=1,3
    IF(SHORT)WRITE BINARY(12)NC ;Space right twice
    IF(SHORT)WRITE BINARY(12)NC ;for short picture
    DO 14 JO=1,N1
    WRITE BINARY(12)IL
7
    WRITE BINARY(12)LPPC ;End with a line feed
15
WRITE BINARY(12)LF
WRITE(12,16)ATITLE(1) ;Title picture
FORMAT(15X,S80) ;Close all channels
CALL RESET
STOP

C Format statements.
C
C 16 FORMAT(’ ',15X,’Signal Processing Laboratory, Air Force
C 1 Institute of Technology, Wright-Patterson AFB, OH 45433<14’)
C 17 FORMAT(S13) ;Filename format
C 18 FORMAT(’0’) ;Double space for short picture
C 19 FORMAT(S1) ;Query format
C 20 ACCEPT" Input error. Try again (YES/NO) > ">
C 21 ACCEPT" Input error. Try again (YES/NO) > ">
C
C Set up parameters for shortened picture.
C
C 22 TYPE" The shortened (noise removed) picture was chosen."
C 20 ACCEPT"Enter starting row (1-256): ",JSTART
C IF(JSTART.LT.1.OR.JSTART.GT.256)G TO 220
C ACCEPT" Number of lines to be displayed (255 Max)? ",N2
C N2=MIN(N2,(257-JSTART))
C SHORT=.TRUE. ;Turn on short test
C N1=63 ;Top and bottom border length
C IF(N2.GT.232)N2=228 ;Number of lines displayed
C N3=3 ;Location of left border
C N4=65 ;Location of right border
C N5=66 ;Location of line feed
C N6=4 ;Length of lines displayed
C IF(JSTART.EQ.1)GO TO 5
C DO 23 J = 1,(JSTART-1) ;Skip first (JSTART-1)
C READ BINARY(1)TRBC
C 23 GO TO 5
C END

C********** Program PICTURE ***********

86
SUBROUTINE WIT3X3 (VIDPIX, LINEPRINT)

Written by Lt. J.H. Cromer

This subroutine converts the video pixel values (i.e. an integer value from 0-15) to lineprinter dot matrix form. A 3x3 array pattern is formed by this subroutine. Dot pattern texture (distribution of dots) and average brightness are varied to create 16 pseudo-gray levels. Odd numbered rows of the picture created by PICTURE use these 3x3 patterns.

NOTE: The six least significant bits of each byte sent to the P-300 represent print hammer switches (i.e. a 1 turns the hammer on to print a dot, a 0 leaves it off). Bit seven must have a value of 1.

(See the Printronix manual for further discussion)

INTEGER VIDPIX, LINEPRINT(3), RIGHT, PATTERN(3,16,2)

Note that right and left pixel patterns are not necessarily the same.

DATA PATTERN / 4*7,5,7,6,7,3,7,5,2*2,7,2,5,2,7,2*2,  
$5,2,5,0,5,2,5,2,5,0,2*2,0,2*2,0,2,1,3*0,2,4*0,  
$120K,150K,120K,170K,2*120K,150K,120K,150K,120K,100K,150K,  
$140K,120K,110K,120K,100K,150K,110K,140K,110K,100K,150K,  
$2*100K,120K,7*100K/  
RIGHT=(VIDPIX.AND.15)+1  
LEFT=(ISHFT(VIDPIX,-4).AND.15)+1  
DO 10 I=1,3  
LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)  
10 CONTINUE  
RETURN  
END

C******** Subroutine OUT3X3 ***************
SUBROUTINE OUT4X3(VIDPIX, LINEPRINT)

C***************************************************************
CWritten by Lt. Cramner
C
This subroutine returns lineprinter pixels to the calling program PICTURE, which sends video pixels (an integer from 0-15). The pixel pattern returned is a 4x3 dot matrix array, to be used for the even rows of pictures created by PICTURE. (See OUT3X3.FR for more explanation).
C***************************************************************

INTEGER VIDPIX, LINEPRINT(4), RIGHT, PATTERN(4,16,2)
DATA PATTERN/5*7,5,7,7,7,3,3,6,7,5,5,6,3,2,5,2,
$3*5,2,2,5,5,0,2,5,2,2,5,2,1,4,2,2,4,1,2,1,4,2,0,0,1,
$4,0,0,2,10*0,5*170K,150K,2*170K,160K,2*170K,2*130K,
$160K,170K,150K,120K,2*170K,120K,150K,120K,3*150K,2*120K,
$150K,120K,2*150K,3*120K,150K,2*120K,110K,140K,120K,150K,
$2*100K,150K,120K,100K,110K,140K,120K,2*100K,
$120K,2*100K,140K,3*100K,120K,5*100K/
RIGHT=(VIDPIX.AND.15)+1
LEFT=(LSHFT(VIDPIX,-4).AND.15)+1
DO 10 I=1,4
     LINEPRINT(I)=PATTERN(I,LEFT,1)+PATTERN(I,RIGHT,2)
10 CONTINUE
RETURN
END
C********************************************************** Subroutine OUT4X3 **********************************************************
This appendix contains the following programs:

1. REMOVE (UNPACK2, TEST3)
2. EVIDHIST (HISTPLOT)
3. NMOVE (TEST, BLOCK, CHANGE)
4. NEGATE
5. REDUCE
6. TONER
Program REMOVED by Lt Jim Cramer

Fortran 5

This program utilizes a 3x3 pixel mask function to remove noise from packed video files. The mask can be changed by modifying the Subroutine TEST3.

Execution Line Format:

REMOVED infile outfile outfile2
All files are 64 block video files.

Load Line Format:

RLDR REMOVED TEST3 UNPACK2 IOF @FLIB@

I/O FILE MANAGEMENT

CALL IOF(3,MAIN,FILE1,FILE2,FILE3,DUM,DUM,DUM,DUM,DUM)
CALL OPEN(1,FILE1,2,IER)
DELETE FILE2
DELETE FILE3
CALL CFILW(FILE2,2,IER)
CALL OPEN(2,FILE2,2,IER)
CALL CFILW(FILE3,2,IER)
CALL OPEN(3,FILE3,2,IER)

Enter 4 threshold values

DO 101 I=1,4
ACCEPT "ENTER THRESH: ",THRESH(I)
TYPE "THRESH("',I',")",".THRESH(I)
101 CONTINUE

Set top border row values

CALL RDBLK(1,0,VIDEO,1,IER)
IF(IER.NE.1)STOP " RDBLK #0 error:",IER
DO 1 I=1,64
OUT2(I)=VIDEO(I)
OUT1(I)=VIDEO(I)
1 CALL UNPACK2(1)
CALL UNPACK2(2)

Set pixel value change counters

DO 10 I=1,16
DO 10 J=1,4
   COUNT(I,J)=0
10 CONTINUE

! Operate on 64 blocks. Each call to TEST3 operates on 3 rows to output 1 row. Four calls to TEST3 are made for each packed block read.

DO 2 I=1,63
   CALL TEST3(A,B,C,2)
   CALL TEST3(B,C,D,3)
   CALL RBLK(1,I,VIDEO,1,IER)
   IF(IER.NE.1) STOP " RBLK error ".I," error:" ,IER
   CALL UNPACK2(1)
   CALL TEST3(C,D,A,4)
   CALL WBLK(2,I-1,OUT1,1,IER)
   IF(IER.NE.1) STOP " WBLK ".I-1," error:" ,IER
   CALL WBLK(3,I-1,OUT2,1,IER)
   CALL TEST3(D,A,B,1)
   CALL UNPACK2(2)
2  TYPE" <15>Block ".I," tested .."

   CALL TEST3(A,B,C,2)
   CALL TEST3(B,C,D,3)

! Set bottom border row values

DO 3 I=193,256
   OUT1(I)=VIDEO(I)
   OUT2(I)=VIDEO(I)
   CALL WBLK(2,63,OUT1,1,IER)
   IF(IER.NE.1) STOP " WBLK ".63," error:" ,IER
   CALL WBLK(3,63,OUT2,1,IER)
3  WRITE *.OUT,1000
   1000 FORMAT( " #" ,I2," $s reduced to","I3","I5"," ",I2,
             "$s increased to ",I2," ",I5)"

20 CONTINUE

WRITE *.OUT,1000
   TYPE" <15>Program REMOVE finished."!
   CALL RESET
   STOP
   END

C********* Program REMOVE ********************************************
SUBROUTINE TEST3(ROW1,ROW2,ROW3,I)

(Called by REMOVE)  by Lt Jim Cromer

This subroutine determines the noise removed
output value of an input pixel by calculating
some function of its 8 nearest neighbors, and
comparing it with a user input threshold. Two
methods of noise removal are used; the output
arrays are OUT1 and OUT2.

Packed video blocks hold 4 rows. The parameter I
determines which part of the block that OUT
will be packed into.

I=1  -->  pack OUT into elements 1-64
I=2  -->  65-128
I=3  -->  129-192
I=4  -->  192-256

* *

INTEGER ROW1(256),ROW2(256),ROW3(256),TEMP1(256)
INTEGER TEMP2(256),OUT1(256),OUT2(256),THRESH(4)
INTEGER COUNT(16,4),COLUMN,SURROUND
LOGICAL TEST
COMMON /COM3/ THRESH,COUNT,OUT1,OUT2

Set border values

TEMP1(1)=ROW2(1)
TEMP2(1)=ROW2(1)
TEMP1(256)=ROW2(256)
TEMP2(256)=ROW2(256)

Check each of the neighbors for a value of
the interior pixel +/-1.

DO 20 COLUMN=2,255

N=1
TEMP1(COLUMN)=ROW2(COLUMN)
TEST=ROW2(COLUMN).EQ.0.OR.ROW2(COLUMN).EQ.15
IF (TEST) GO TO 20
I2=ROW2(COLUMN)-3
J=0
I2=I2+2
DO 5 M=(COLUMN-1),(COLUMN+1)

IF (ROW1(M).EQ.I2) J=J+1
IF (ROW2(M).EQ.I2) J=J+1
IF (ROW3(M).EQ.I2) J=J+1
CONTINUE

TEST=J.GE.THRESH(N)
IF (TEST) TEMP1(COLUMN)=I2
IF (TEST) COUNT(I2,N)=COUNT(I2,N)+1
IF (TEST) GO TO 20

92
N=N+1
IF (N.EQ.2) GO TO 1
CONTINUE

Compare the average value of the surround less the interior value with a threshold.

DO 100 COLUNM=2,255
TEMP2 (COLUNM)=ROW2 (COLUNM)
TEST=ROW2 (COLUNM).EQ.0.OR.ROW2 (COLUNM).EQ.15
IF (TEST) GO TO 100
SURROUND=1*ROW2 (COLUNM)
DO 50 M=(COLUMN-1),(COLUMN+1)
SURROUND=SURROUND+ROW1 (COLUMN)+ROW2 (COLUMN)
50 CONTINUE

AVERAGE=FLOAT (SURROUND)/8.0
DIFF=AVERAGE-FLOAT (ROW2 (COLUNM))
I2=ROW2 (COLUNM)+1
I3=INT (AVERAGE+0.5)
I4=(ROW2 (COLUNM)+I3)/2
I5=MAX (I4,12)
I6=MIN (I4,12-2)
TEST=DIFF.GE.THESSH (3)
IF (TEST) TEMP2 (COLUMN)=I5
IF (TEST) COUNT (I5,3)=COUNT (I5,3)+1
IF (TEST) GO TO 100
TEST=(-1*DIFF).GE.THESSH (4)
IF (TEST) TEMP2 (COLUMN)=I6
IF (TEST) COUNT (I2-2,4)=COUNT (I2-2,4)+1
100 CONTINUE

Pack the noise removed pixels to OUT1, OUT2

M=3
JLOW=(I-1)*64+1
JHIGH=JLOW+63
DO 200 J=JLOW,JHIGH
M=M+4
OUT1 (J)=ISHFT (TEMP1 (M),12)+ISHFT (TEMP1 (M+1),8)+
ISHFT (TEMP1 (M+2),4)+TEMP1 (M+3)
OUT2 (J)=ISHFT (TEMP2 (M),12)+ISHFT (TEMP2 (M+1),8)+
ISHFT (TEMP2 (M+2),4)+TEMP2 (M+3)
200 CONTINUE
RETURN
END

C******** Subroutine TEST3 ***************************************
SUBROUTINE UNPACK2(I)

(Called by REMOVE) by Lt. Jim Cromer

If I=1 ---> The first two rows of a packed block are unpacked into arrays A and B.
Else ---> The last two rows are unpacked into arrays C and D.

(See the unpacking subroutines for further explanation).

INTEGER A(256), B(256), C(256), D(256), VIDEO(256)
COMMON /COM2/A, B, C, D, VIDEO

IF(I .EQ. 1) GO TO 3

L=3
DO 2 K=129,192
   I=I+4
   C(L) = 15 .AND. ISHFT(VIDEO(K), -12)
   C(L+1) = 15 .AND. ISHFT(VIDEO(K), -8)
   C(L+2) = 15 .AND. ISHFT(VIDEO(K), -4)
   C(L+3) = 15 .AND. VIDEO(K)
   D(L) = 15 .AND. ISHFT(VIDEO(K+64), -12)
   D(L+1) = 15 .AND. ISHFT(VIDEO(K+64), -8)
   D(L+2) = 15 .AND. ISHFT(VIDEO(K+64), -4)
   D(L+3) = 15 .AND. VIDEO(K+64)

2 CONTINUE
   GO TO 6

C

3 L=3
DO 5 K=1,56
   I=I+4
   A(L) = 15 .AND. ISHFT(VIDEO(K), -12)
   A(L+1) = 15 .AND. ISHFT(VIDEO(K), -8)
   A(L+2) = 15 .AND. ISHFT(VIDEO(K), -4)
   A(L+3) = 15 .AND. VIDEO(K)
   B(L) = 15 .AND. ISHFT(VIDEO(K+64), -12)
   B(L+1) = 15 .AND. ISHFT(VIDEO(K+64), -8)
   B(L+2) = 15 .AND. ISHFT(VIDEO(K+64), -4)
   B(L+3) = 15 .AND. VIDEO(K+64)

5 CONTINUE

6 RETURN
END

C ******** Subroutine UNPACK2 ****************************
Program EVIDHIST Written by Lt. Jim Cromer
Fortran 5

This program calculates a histogram of the 16 gray levels of the input video picture, and types the results on the CRT and/or lineprinter in tabular form. A plot can be sent to the lineprinter if requested.

Execution Line Format: (run VIDHIST on the NOVA)
EVIDHIST

Load Line Format:
MLRM EVIDHIST UNPACK HISTPLOT PLOT5.LB @LIB@
A link to PLOT5.LB in DP5F:CALCOM5 should exist or be created before loading.

REAL VALUE(0:15)
INTEGER INFILE(7),IARRAY(2048),IUNPACK(8192)

*** I/O FILE MANAGEMENT ***********************************************

1 ACCEPT "Enter name of video file to be evaluated $ --> "
READ(11,1000)INFILE(1)
1000 FORMAT(S13)
CALL OPEN(1,INFILE,1,IER)
IF(IER.NE.1)STOP "INFILE OPEN error ",IER

*** INITIALIZE DATA ***********************************************

DO 10 I=0,15
   VALUE(I)=0.0
10 CONTINUE

*** PROCESS THE PICTURE ***********************************************

KK=8 ;This variable determines the # of blocks
      ;read at a time. (Maximum=16, if array sizes are increased.)

LL=64-KK
DO 100 J=0,LL,1
   CALL ROBLK(1,J,IARRAY,KK,IER)
   IF(IER.NE.1)TYPE "ROBLK #",J," error: ",IER
   ISIZE=256*KK ;256 words/block
   CALL UNPACK(ISIZE,IARRAY,IUNPACK)
   IMAX=ISIZE*4 ;4 pixels/word
   DO 50 I=1,IMAX
      IPIX=IUNPACK(I)
      VALUE(IPIX)=VALUE(IPIX)+1.0
50 CONTINUE
C
C***** Print the histogram ********
C
100 CONTINUE
   TYPE"<15>"*
   TYPE"Where should the histogram table be printed?"
   TYPE"Enter 1 to send to lineprinter, 2 to send to CRT,"
   TYPE"or 3 for both; any other integer to continue."
   ACCEPT"<11><11>Enter integer option --- > ",ILOPT
   TYPE"<15>"*
   IF (ILOPT.LT.1.OR.ILOPT.GT.3) GO TO 160
   IF (ILOPT.EQ.1.OR.ILOPT.EQ.3) ICH=12
   110 IF (ILOPT.EQ.2) ICH=10
       IF (ICH.EQ.10) WRITE(10,5000) INFILE(1)
       IF (ICH.EQ.12) WRITE(ICH,2000) INFILE(1)
       DO 120 J=0,7
           JJ=J+8
           WRITE (ICH,3000) J, VALUE(J), JJ, VALUE(JJ)
       CONTINUE
       IF (ICH.EQ.12) WRITE(12,4000)
   IF (ILOPT.EQ.3) ILOPT=2
   IF (ILOPT.EQ.2.AND.ICH.EQ.12) GO TO 110
   160 TYPE"<15>"*

C
C***** Plot the histogram ********
C
   TYPE"Enter 1 to plot histogram on lineprinter, any "
   ACCEPT"other integer to continue: ",I
   IF (I.NE.1) GO TO 200
   CALL HISTPLOT(VALUE, INFILE)

C
C***** EXIT program or start over ********
C
200 CALL RESET
   2000 FORMAT(//77X," <10>HISTOGRAM OF ",S13,",/25X,
       "$" _______________//)
   3000 FORMAT(18X, " # of level","I2,"":"I5", " # of level","I3","":"I5)
   4000 FORMAT("0")
   5000 FORMAT(//27X,"HISTOGRAM OF ",S13,/
       TYPE"<15>"
   ACCEPT"Enter 1 to evaluate another file, any other integer
   $ to stop: ",I
   IF (I.EQ.1) GO TO 1
   STOP
   END

C
C*********** Program EVIDHIST ****************************
SUBROUTINE HISTPLOT(VALUE, INFILE)
C*****************************************************************************
C Written by Lt. Jim Cramner
C This subroutine plots the 16 element array VALUE
C passed to it as a histogram. It is called by the
C program EVIDHIST.
C*****************************************************************************
REAL VALUE(0:15), INFILE(7)
REAL X(481), Y(481)
K=1
X(1)=-0.25 ; X-axis starting point
C******** Create the arrays to be plotted **********************
C
VNDF=655.36 ; # pixels/100
1 ACCEPT Enter histogram level to be suppressed
$ (999 to continue): ', ISUPPRESS
IF (ISUPPRESS.EQ.999) GO TO 3
IF (ISUPPRESS.LT.0 .OR. ISUPPRESS.GT.15) TYPE "INPUT ERROR! <7><15>"
IF (ISUPPRESS.LT.0 .OR. ISUPPRESS.GT.15) GO TO 1
VNDF=VNDF.VALUE(ISUPPRESS)/100.0
VALUE(ISUPPRESS)=0.0
GO TO 1
3 DO 20 I=0,15 ; do 16 values
   DO 5 J=1,15 ; determines width of bars
      K=K+1
      X(K)=X(K-1)+1.0/30.0
      Y(K)=VALUE(I)/VNDF
      CONTINUE
5 X(K-15)=X(K-14)
   DO 10 J=16,30 ; determines spacing between bars
      K=K+1
      X(K)=X(K-1)+1.0/30.0
      Y(K)=0.0
10 CONTINUE
C Find end of input filename, then insert blanks after it
C
DO 30 J=2,13
   IF ((BYTE(INFILE,J)).EQ.0) BYTE(INFILE,J)=32
   IF ((BYTE(INFILE,J-1)).EQ.32) BYTE(INFILE,J)=32
30 CONTINUE
Y(480)=0.0
X(480)=16.0
XAX=5.0 ; set X- and Y-axes length
YAX=4.0
C******** Plot the arrays **********************************************
C
TYPE "Please wait while a plot is generated (50 secs)."
CALL PLOTS(0,0,6)
CALL PLOT(1.5,5.0,-3)
CALL ASCALE(X,XAX,480,1,FX,DX) ;scale the arrays
CALL ASCALE(Y,YAX,480,1,FY,DY)

C. Title the axes
C
CALL AXIS(0.0,0.0,"PIXEL VALUE",-11,XAX,0.0,FX,DX)
CALL AXIS(0.0,0.0,"PERCENTAGE OF EVALUATED $\$\$ P I X E L S$",30,YAX,90.0,FY,DY) ;vertical axis title
CALL ALINE(X,X,480,1,0,1,FX,DX,FY,DY)

C. Title the plot
C
CALL SYMBOL(0.1,4.0,0.2,"HISTOGRAM OF ",0.0,13)
CALL SYMBOL(3.1,4.0,0.2,INFILE,0.0,13)
CALL PLOT(0.0,0.0,999) ;send to lineprinter
RETURN
END

C************ Subroutine HISTPLOT **********************
This program will convert individual pixel values of an input video file into new values assigned by the user. It can be used to adjust gray-levels (i.e. histogram equalization), increase contrast, display selected pixel values, create "negative images", create files of a constant gray-level, and for many other purposes. (NOTE: The program EVIDHIST can be run to generate a histogram of any VIDEO file.)

Execution Line Format:
	TONER
The program will ask for the input and output file names, and the new pixel values.

Load Line Format:
	RLDR TONER TIMER UNPACK REPACK @FLIB@

C***************************************************************
C
C Program TONER     Written by Lt. Jim Cromer
C
C This program will convert individual pixel values of an input video file into new values assigned by the user. It can be used to adjust gray-levels (i.e. histogram equalization), increase contrast, display selected pixel values, create "negative images", create files of a constant gray-level, and for many other purposes. (NOTE: The program EVIDHIST can be run to generate a histogram of any VIDEO file.)
C
C Execution Line Format:
C           TONER
C    The program will ask for the input and output file names, and the new pixel values.
C
C Load Line Format:
C             RLDR TONER TIMER UNPACK REPACK @FLIB@
C
C***************************************************************
C
INTEGER NEWVALUE(0:15),INFILE(7),OUTFILE(7)
INTEGER PACKED(4096),UNPACKED(16384)
ISTART=0 ;start timer
ISTOP=1  ;stop timer

C******** USER INPUT OF VARIABLES ********************

ACCEPT"Enter name of video file to be toned ----> "
READ(11,1000)INFILE(1)
ACCEPT"Enter name of output file ----> "
READ(11,1000)OUTFILE(1)
TYPE"<15>""Enter new pixel values<15>"
TYPE"NOTE: Leading zeros are significant, i.e. enter"
TYPE" a '1' as '01', enter a '2' as '02', etc.<15>"
DO 1 J=0,15 ;16 gray-levels
WRITE(10,2000)J
READ(11,3000)NEWVALUE(J)
IF (NEWVALUE(J).LT.0.OR.NEWVALUE(J) .GT. 15)
$NEWVALUE(J)=15
1 CONTINUE
CALL TIMER(ISTART)

C******** I/O FILE MANAGEMENT ********************

CALL OPEN(1,INFILE,1,IER)
IF(IER.NE.1) TYPE"INFILE OPEN error $",IER
CALL DFILW(OUTFILE,IER)
IF(IER.NE.1.AND.IER.NE.13) TYPE"OUTFILE DFILW error $",IER
$ error $,IER
CALL DFILW(OUTFILE,3,64,IER) ;create a contiguous file
IF(IER.EQ.41) CALL DFILW(OUTFILE,2,IER)
IF (IER .NE. 1) TYPE "OUTFILE CPILW error ", IER
CALL OPEN (2, OUTFILE, 3, IER)
IF (IER .NE. 1) TYPE "OUTFILE OPEN error ", IER

C ******** RE-TONE THE PICTURE *****************************************
C
DO 3 J = 0, 48, 16
   CALL RDBLK (1, J, PACKED, 16, IER)
   IF (IER .NE. 1) TYPE "RDBLK ", J, " error ", IER
   CALL UNPACK (4096, PACKED, UNPACKED)
   DO 2 I = 1, 16384 ; do 1/4 of picture
      UNPACKED (I) = NEWVALUE (UNPACKED (I))
   2 CONTINUE
   CALL RREPACK (4096, UNPACKED, PACKED)
   CALL WRBLK (2, J, PACKED, 16, IER)
   IF (IER .NE. 1) TYPE "WRBLK ", J, " error ", IER
3 CONTINUE
C
Send message to CRT terminal
C
CALL TIMER (ISTOP)
WRITE (10, 4000) OUTFILE (1)
TYPE "<15>, "Have a nice day!<7><15>"
C
C ******** WRITE NEW TONE VALUES TO THE LINEPRINTER ************
C
WRITE (12, 5000)
WRITE (12, 6000) INFILE (1), OUTFILE (1)
DO 5 I = 0, 15
   WRITE (12, 7000) I, NEWVALUE (I)
5 CONTINUE
1000 FORMAT (S13)
2000 FORMAT (" Change old pixel value", I3, " to ")
3000 FORMAT (I2)
4000 FORMAT (" The toned picture is in the file ——> ", $S13)
5000 FORMAT ("//////\Z5X," RESULTS OF TONER<10>"
9$26X," \\
6000 FORMAT (10X, " Input file ——> ", S13, /10X, " Output file
S<10>, S13, /20X, "OLD PIXEL", 10X, "NEW PIXEL", /20X,
9$10X, "")
7000 FORMAT (23X, I2, 17X, I2)
CALL RESET
STOP
END
C
C ******** Program TONER ***********************************************
Program NMOVE Written by Lt. Jim Crcffer
Fortran IV 16 Aug 1982

This program will place the video information in the window given for the template (INFILE1) inside of the window given for the background (INFILE2), and write the combined picture to the output file. The window may be placed anywhere within the background, and may be taken from anywhere within the template. Window width, length, and position are input by the user.

Execution line format: (on the NOVA only)
NMOVE
(run NMOVE on the ECLIPSE)

Loader command line format (NOVA only):
RLDR NMOVE TEST BLOCK CHANGE XRDBLK XRDBLK
UNPACK REPACK FORT,LB

********** I/O FILE MANAGEMENT ***********************

TYPE"<15>" "$Program NMOVE is to be run on the NOVA only!"
TYPE"<15>" "$********** I/O FILE MANAGEMENT ***************<15>"
ACCEPT" Enter template file name: "
READ(11,1000) INFILE1(1)
ACCEPT" Enter background file name: "
READ(11,1000) INFILE2(1)
DO 999 J=1,7
999 OUTFILE(J)=INFILE2(J)
ACCEPT" Enter combined output file name: "
READ(11,1000) OUTFILE(1)
1000 FORMAT(S13)
CALL OPEN(1,INFILE1,2,IER)
IF(IER.NE.1)TYPE" Channel 1 OPEN error: ",IER
CALL OPEN(2,INFILE2,2,IER)
IF(IER.NE.1)TYPE" Channel 2 OPEN error: ",IER
CH3=2
ICOUNT=0
DO 1002 J=1,7 ;check for BACKGROUND=COMBINED
1002 IF(OUTFILE(J).EQ.INFILE2(J))ICOUNT=ICOUNT+1
IF(ICH3.EQ.7)GO TO 1
CH3=3
CALL DFILM(OUTFILE,IER)
IF (IER.NE.1 .AND. IER.NE.13) TYPE"OUTFILE DFILW error:",IER
CALL CPFILE(OUTFILE,2,IER)
IF (IER.NE.1) TYPE" CFILW error:",IER
CALL OPEN(CH3,OUTFILE,2,IER)
IF (IER.NE.1) TYPE" Channel 3 OPEN error:",IER

C******* ENTER WINDOW PARAMETERS ***********************************

ACCEPT"<15>"," Enter top row of template window (1-256):",TTOP
ACCEPT" Enter left column of template window (1-256):",TLEFT
ACCEPT" Enter width of window (1-256):",WIDTH
ACCEPT" Enter length of window (1-256):",LENGTH

The calls to TEST check to see if the input parameters are
legal, and modifies them if necessary:
0 < TOP < 257, (TOP + LENGTH) < 258
0 < LEFT < 257, (LEFT + WIDTH) < 258

CALL TEST(TTOP,TLEFT)
ACCEPT"<15>"," Enter top row of background window (1-256):",CTOP
ACCEPT" Enter left column of background window (1-256):",CLEFT
CALL TEST(CTOP,CLEFT)
CALL BLOCK(TBLOCKS,TBL,TLS,TCOL,TTOP,TLEFT)
CALL BLOCK(CBLOCKS,CBL,CLS,CCOL,CTOP,CLEFT)

Determine column number of the last video row (0-3)

J1=MOD(LENGTH,4)
TSTOP=MOD(TCOL+J1),4)-1
CTSTOP=MOD(TCOL+J1),4)-1
IF (CTSTOP.EQ.-1) CTSTOP=3
IF (TSTOP.EQ.-1) TSTOP=3

Determine the last significant block of window

CLB=CBL+CBLOCKS-1
TBL=TBL+TBLOCKS-1

User check of window parameters

TYPE"<15>"," WIDTH=""WIDTH"," LENGTH=""LENGTH
TYPE"TEMPLATE TOP ROW=""TTOP"," BACKGROUND TOP ROW=""CTOP
TYPE"TEMPLATE LEFT COLUMN=""TLS"," BACKGROUND LEFT COLUMN
$=""CLS","<15>"
ACCEPT"Enter 1 to see expanded set of variables, any
$ other integer to continue: ",I
IF (I.NE.1) GO TO 5

TYPE************************************************************
TYPE" PARAMETER TEMPLATE BACKGROUND"
TYPE"<15>"," TOP ROW ",TTOP,CTOP
TYPE"<15>"," START COL ",TCOL,CCOL
TYPE"<15>"," STOP COL ",TSTOP,CSTOP
TYPE"<15>"; "FIRST BLOCK",TBl,CH1
TYPE"<15>"; "LAST BLOCK",TBl,CH1
TYPE"<15>"; "OF BLOCKS",TBl,CH1,SBLOCKS
TYPE"<15>"; "LEFT COL",TBl,CH1,SBLOCKS
TYPE"<15>"; "WIDTH="; WIDTH
TYPE"<15>"; "LENGTH="; LENGTH
TYPE"<15>"; "******************************************************************
ACCEPT" Enter 1 to try another set, any other integer
$ to continue: ",I
IF(I.NQ.1)GO TO 1

C
C***** Create the combined picture ***************
C
5 ICOUNT=0
IF(CH3.EQ.2)GO TO 20 ;If combined picture file
; is the same as the back-ground picture file, then no
; need to write to itself
C
Write background only blocks (before window)
to the combined picture file.
C
JMAX=CH1-1
IF(JMAX.LT.0)GO TO 20
DO 10 J=0,JMAX
   CALL ROBLK(2,J,BACK,1,IER)
   IF(IER.NE.1)TYPE" 2ROBLK",J," error:",IER
   CALL WRBLK(CH3,J,COMB,1,IER)
   IF(IER.NE.1)TYPE" WRBLK",J," error:",IER
   ICOUNT=ICOUNT+1
10 CONTINUE
20 TYPE" Background before window completed."
   TYPE" # Blocks written:";ICOUNT
C
C ... Overlay template window onto background ........
C
CALL XRDBLK(1,TBl,TEMP,1,IER)
IF(IER.NE.1)TYPE"XRDBLK ",TBl," error:",IER
CALL XRDBLK(2,CBl,BACK,1,IER)
IF(IER.NE.1)TYPE"XRDBLK ",CBl," error:",IER
NL=TOOL ;4-MAX(NL,N2) gives the number of rows
N2=COOL ;to change before the next RDBLK
IF(TOOL.GT.COOL)GO TO 100
C
C
C
There are four columns in the packed video array (64x4),
designated 0, 1, 2, and 3. If the template starting
column number is less than or equal to the background (combined)
starting column number, then the background block will be "used
up" before the template block. When the background block is
finished, a WRBLK is done, and the next background block is read
When the template block is finished, the next template block is
read, but no WRBLK needs to be performed. Note that the back-
ground and combined files are always at the same block number.

CALL CHANGE(N2,N2,N1)
CALL XRDBLK(CH3,CBl,COMB,1,IER)
IF(IER,NE.1) Type" WRBLK ",CBl," error: ",IER

Write the template window into the background

TB=TB1+1
IMIN=CBl+1
ICOUNT=1
IMAX=CB1-1
IF(IMIN.GT.IMAX) GO TO 60
DO 50 I=IMIN,IMAX
    CALL XRD_BLK(2,I,BACK,1,IER)
    IF(IER,NE.1) Type" 2RBLK ",I," error: ",IER
    CALL CHANGE(N1,N2,N1)
    CALL XRDBLK(1,TT,TEMP,1,IER)
    IF(IER,NE.1) Type" 1RBLK ",TB," error: ",IER
    CALL CHANGE(N2,N2,N1)
    CALL XRDBLK(CH3,1,COMB,1,IER)
    IF(IER,NE.1) Type" WRBLK ",I," error: ",IER
    ICOUNT=ICOUNT+1
50    TB=TB1+1
60    Type" 0COL,IT,0COL-Window portion complete."
    Type" # blocks written: ",ICOUNT
    GO TO 250

In this case the template starting column number
is greater than the background starting column number.
The template block must be "finished" first.

CALL CHANGE(N1,N2,N1) ;finish TEMP block
TB=TB1+1
IMAX=CB1-1
ICOUNT=0
IF(CBl.GT.IMAX) GO TO 225
DO 200 I=CBl,IMAX
    CALL XRDBLK(1,TT,TEMP,1,IER)
    IF(IER,NE.1) Type" 1RBLK ",TB," error: ",IER
    CALL CHANGE(N2,N2,N1) ;finish BACK block
    CALL XRDBLK(CH3,1,COMB,1,IER)
    IF(IER,NE.1) Type" WRBLK ",I," error: ",IER
    ICOUNT=ICOUNT+1
    IBLK=I+1
    CALL XRDBLK(2,IBLK,BACK,1,IER)
    IF(IER,NE.1) Type" 2RBLK ",IBLK," error: ",IER
    CALL CHANGE(N1,N2,N1) ;finish TEMP block
200    TB=TB1+1
225    Type" 0COL,IT,0COL-Window portion complete."
    Type" # blocks written: ",ICOUNT
If the combined (background) stopping column number is greater than the template stop column number, then the second last template block (121B-TLB-1) must be read (i.e. there are more video rows to be changed in the last background block than there are available in the last template block to change them to). If TSTOP is greater than or equal to CSTOP, then there are sufficient video rows available in the last template block to complete the last background block to be changed.

IF(CSTOP.GT.TSTOP)GO TO 400
M1=TSTOP-CSTOP
CALL XRDBLK(1,TLB,TEMP,1,IER)
IF(IER.NE.1)TYPE" 1RDBLK ",TLB," error:",IER
CALL XRDBLK(2,CLB,BACK,1,IER)
IF(IER.NE.1)TYPE" 2RDBLK ",CLB," error:",IER
N1=3-CSTOP
N2=0
CALL CHANGE(N1,N2,N1) ;finish BACK block to CSTOP
CALL XRDBLK(CH3,CLB,COMB,1,IER)
IF(IER.NE.1)TYPE" WREBLK ",CLB," error:",IER
TYPE" CSTOP,IP,TEMP—Last block of window complete."
GO TO 500

Complete the last block of the window
NOTE: CSTOP is greater than TSTOP. Therefore finish TEMP before BACK.

400 M1=CSTOP-TSTOP
CALL XRDBLK(2,CLB,BACK,1,IER)
IF(IER.NE.1)TYPE" 2RDBLK ",CLB," error:",IER
121B-TLB-1
CALL XRDBLK(1,121B,TEMP,1,IER)
IF(IER.NE.1)TYPE" 1RDBLK ",121B," error:",IER
N1=4-M1
N2=0
CALL CHANGE(N1,N2,N1) ;finish TEMP block
CALL XRDBLK(1,TLB,TEMP,1,IER)
IF(IER.NE.1)TYPE" 1RDBLK ",TLB," error:",IER
M1=3-TSTOP
CALL CHANGE(N1,N2,N1) ;finish BACK block to CSTOP
CALL XRDBLK(CH3,CLB,COMB,1,IER)
IF(IER.NE.1)TYPE" WREBLK ",CLB," error:",IER
TYPE" CSTOP,GT,TSTOP—Last block of window complete."
ICOUNT=1
500 Finish the combined file (background only portion)

JMIN=CLB+1
IF(JMIN.GT.63)GO TO 601 ;if finished STOP
IF(CH3.EQ.2)GO TO 601 ;if COMBINED=BACKGROUND, STOP
DO 600 J=JMIN,63
CALL XRDBLK(2,J,BACK,1,IER)

105
IF(IER.NE.1) TYPE" 2DDBLK #"J," error:"IER
CALL WRBLK(CH3,J,COMB,1,IER)
IF(IER.NE.1) TYPE" WRBLK #"J," error:"IER
ICOUNT=ICOUNT+1

600 CONTINUE
TYPE" Finished background only portion."
601 TYPE" # blocks written:"ICOUNT
TYPE"<15>"",<7>"<15>" Program NMOVE execution completed. <7>
WRITE(10,2000)OUTFILE(1)
2000 FORMAT(" The combined picture is in the file --> ",Sl3)
C
C
C***** Present Option Menu ************************************
C
GO TO 2010
2002 TYPE"<15>"","Input error. Try again."
2010 TYPE"<15>"",********************************************
TYPE"<15>"",<11>1 - Try another set of window values"
TYPE"<15>"",<11>2 - Start over with new input pictures"
C
TYPE"<15>"",<11>3 - Display combined picture on the video monitor"
TYPE"<15>"",<11>3 - Save combined picture and STOP<15>
ACCEPT"<11>Enter option --> "IOPT
IF(IOPT.LT.1 .OR. IOPT.GT.3)GO TO 2002
IF(IOPT.EQ.1)GO TO 1
CALL RESET
IF(IOPT.EQ.2)GO TO 99
IF(IOPT.EQ.3)STOP
TYPE"<15>"",Check monitor -- Press green CHOPS control
button to continue."
IDCNT=4
IPAR(1)=9999
IPAR(2)=0
WRITE(10,3000)OUTFILE(1)
3000 FORMAT("0","Picture being displayed --> ",Sl3)
C
CALL CHANNEL(0,0,0,0,"A",0,0,0,IPAR,ISYS)
C
CALL CHANNEL(3,1,2,1,ICOUNT,OUTFILE,64,0,IPAR,IERR,ISYS)
C
CALL ERCHR(IERR,1,ICOUNT,1,ISYS)
TYPE"<15>"",CHANNEL currently not loaded."
TYPE"Use VIDEO to display combined pictures.<15>"
CALL OPEN(1,INFILE1,2,IER)
IF(IER.NE.1) TYPE"CH1 RE-OPEN ERROR:"IER
CALL OPEN(2,INFILE2,2,IER)
IF(IER.NE.1) TYPE"CH2 RE-OPEN ERROR:"IER
IF(CH3.EQ.3)CALL OPEN(3,OUTFILE,2,IER)
IF(IER.NE.1) TYPE"CH3 RE-OPEN ERROR:"IER
GO TO 2010
END
C
C***** Program NMOVE ************************************
**SUBROUTINE TEST(TOP,LEFT)**

C************************************************************************************
C Subroutine TEST checks to see if the input parameters
to program NMOVE are legal, and modifies them if
necessary. (It is also called by DISTANCE.)
C************************************************************************************

INTEGER TOP,WIDTH
COMMON/LIST2/LENGTH,WIDTH
IF(LEFT.LT.1.OR.LEFT.GT.256)LEFT=1
MAXWIDTH=257-LEFT ;picture has 256 columns
IF(WIDTH.GT.MAXWIDTH,OR.WIDTH.LT.1)WIDTH=MAXWIDTH
IF(TOP.LT.1.OR.TOP.GT.256)TOP=1
MAXLENGTH=257-TOP ;picture has 256 rows
IF(LENGTH.GT.MAXLENGTH,OR.LENGTH.LT.1)LENGTH=MAXLENGTH
RETURN
END

C********** Subroutine TEST ********************
SUBROUTINE BLOCK(NUMBLOCKS,BLOCK1,LEFTSIDE,COLUMN, TOP, LEFT)
C*****************************************************************************
C Subroutine BLOCK determines the total number of blocks to be read into the window, the first block to be read, and the first video row "column" number. This subroutine is called by NMOVE.
C*****************************************************************************
INTEGER BLOCK1,COLUMN, TOP, REMAINDER, WIDTH
COMMON/LIST2/LENGTH, WIDTH
BLOCK1=INT((TOP-1)/4.0) ; 4 rows per block
COLUMN=MOD((TOP-1),4)
LEFTSIDE=LEFT
REMAINDER=MOD(LENGTH,4)
KL=LENGTH+3
NUMBLOCKS=INT(KL/4.0)
IF (REMAINDER.EQ.2.AND.COLUMN.GT.2) NUMBLOCKS=NUMBLOCKS+1
IF (REMAINDER.EQ.3.AND.COLUMN.GT.1) NUMBLOCKS=NUMBLOCKS+1
IF (REMAINDER.EQ.0.AND.COLUMN.GT.0) NUMBLOCKS=NUMBLOCKS+1
IF (NUMBLOCKS.GT.1) RETURN
TYPE=WARNING: # Blocks to be read =",NUMBLOCKS
PAUSE
RETURN
END
C***************************************************************************** Subroutine BLOCK *****************************************************************************
SUBROUTINE CHANGE(JMIN,CSTART,TSTART)

C************************************************************************************************
C
C Written by Lt. Jim Cromer
C
C Subroutine CHANGE changes the corresponding background
C (i.e. the combined picture) pixels to template pixels;
C it is called by the program NMOVE.
C
C************************************************************************************************

INTEGER COMB(1024), TEMP(1024), CLS, TLS, CSTART, TSTART, WIDTH
COMMON /LIST2/ LENGTH, WIDTH
COMMON /LIST1/ COMB, TEMP, CLS, TLS

C DO 2 J=JMIN,3
   K=TSTART*256+TLS ; Set left side of input (template
   M=CSTART*256+CLS ; and output (combined) windows.
   KMAX=K+WIDTH-1 ; Change values over the width of window
   DO 1 L=K, KMAX
       COMB(M)=TEMP(L)
       M=M+1

1 TSTART=TSTART+1
   CSTART=CSTART+1
   IF (CSTART.EQ.4) CSTART=0
   IF (TSTART.EQ.4) TSTART=0
   RETURN

END

C************************************************************************* Subroutine CHANGE *****************************************************
Program NEGATE by Lt. Jim Cromer
Fortran 5

This program writes to an output video file the "negative" pixels values of the input video file (i.e. dark pixels become light, and vice-versa). It will also "flip" the picture about its horizontal axis (i.e. turn it upside down).

Execution Line Format:
NEGATE[/F] infile outfile

The /F is chosen if the picture is to be reversed horizontally. The program can be run twice to produce a horizontally flipped positive image.

Load Line Format:
RLDR NEGATE IOF UNPACK REPACK TIMER @FLIB@

***************************************************************
INTEGER INFILE(7),OUTFILE(7),GS(2),NEW(1024),OLD(1024)
INTEGER VIDEO(256),DUM,MAIN(7)
***************************************************************

CALL IOF(2,MAIN,INFILE,OUTFILE,DUM,GS,DUM,DUM,DUM)
CALL TIMER(0) ;start timer
OPEN 1,INFILE
DELETE OUTFILE
CALL CFILW(OUTFILE,3,64,IER)
IF(IER.EQ.1) TYPE "Contiguous file created"
IF(IER.EQ.41)CALL CFILW(OUTFILE,2,IER)
IF(IER.NE.1)STOP " Random file creation error"
OPEN 2,OUTFILE

******** Test switch and set variables *********************

 IF(GS(1).EQ.1024)GO TO 1 ;test global switch
 TYPE " Creating negative video file only."
  I1=0
  I2=256
  I3=512
  I4=768
  GO TO 2
1 TYPE " Creating negative and horizontally flipped video file."
  I1=768
  I2=512
  I3=256
  I4=0

******** Loop around the next section 64 times to process the entire picture ************
DO 4 I=0,63
   IF(I.EQ.768) K=63-I
   CALL RDGBK(1,K,VIDEO,1,IER)
   IF(IER.NE.1) TYPE " RDGBK error ",IER STOP
   CALL UNPACK(256,VIDEO,OLD)

Negate pixels, and re-arrange if required.

Work on 4 rows at a time.

DO 3 N=1,256
   NEW(N)=15-OLD(N+1)
   NEW(N+256)=15-OLD(N+12)
   NEW(N+512)=15-OLD(N+13)
   NEW(N+768)=15-OLD(N+14)
CONTINUE

CALL REPACK(256,NEW,VIDEO)
CALL WRBLK(2,1,VIDEO,1,IER)
IF(IER.NE.1) STOP " WRBLK error ",IER
CONTINUE

******************************************************************************

CALL RESET
CALL TIMER(1000)
WRITE(10,5)OUTFILE(1)

FORMAT(" The negative picture is in the file --> ",S13)
STOP
END

******************************************************************************

Program NEGATE
Program REDUCE
Written by Lt. Jim Cramer

This program reduces a 256x256 pixel video picture into a 128x128 array by averaging 4 pixels into 1. The reduced array is placed in the upper left-hand quadrant; all other pixels in the output picture are made white for display purposes.

Execution Line Format:
   REDUCE infile outfile
   Both the input and output files are 256x256 video files (i.e. packed integer form, 64 blocks per file).

Load Line Format:
   RLDR REDUCE IOF TIMER UNPACK REPACK @FLIB@

INTEGER OLDPACKED(4096), OLDPACK(16384), NEWUNPACK(8192)
INTEGER NEWPACKED(2048), INFILE(7), OUTFILE(7), MAIN(7)
COMMON OLDPACKED, OLDPACK
EQUIVALENCE (OLDPACK, NEWUNPACK), (NEWPACKED, OLDPACKED)

******** I/O FILE MANAGEMENT ********

CALL IOF(2, MAIN, INFILE, OUTFILE, I1, I2, I3, I4, I5)
CALL TIMER(0) ; start timer
CALL DFILW(OUTFILE, IER)
IF (IER.NE.1) TYPE"OUTFILE DFILW error: ", IER
CALL CFILW(OUTFILE, 3, 64, IER)
IF (IER.LE.41) CALL CFILW(OUTFILE, 2, IER)
IF (IER.NE.1) TYPE"OUTFILE CFILW error: ", IER
CALL OPEN (1, INFILE, 1, IER)
IF (IER.NE.1) TYPE"INFILW OPEN error: ", IER
CALL OPEN (2, OUTFILE, 3, IER)
IF (IER.NE.1) TYPE"OUTFILE OPEN error: ", IER

******** Process the picture ********

DO 400 M=0,3
   ML=M+16 ; RDBLK counter
   CALL RDBLK(1, ML, OLDPACKED, 16, IER); read 64 rows
   IF (IER.NE.1) TYPE"RDBLK # ML, " error: ", IER
   CALL UNPACK(4096, OLDPACKED, OLDPACK)
   K=0 ; new row counter

This section reduces 64 rows of 256 elements each into 32 rows of 128 elements each. It executes this function 4 times so that a total of 256 rows are reduced to 128.

DO 300 J=1,32 ; create 32 new rows from 64
DO 100 I=1,128 ;create 128 elements/new row
   K=K+1
   L=L*2 ;old row 1 counter
   LL=L+256 ;old row 2 counter
   A1=OLDUNPACK(L-1)
   A2=OLDUNPACK(L)
   A3=OLDUNPACK(LL-1)
   A4=OLDUNPACK(LL) ;average the 4 pixels
   NEWUNPACK(K)=IFIX((A1+A2+A3+A4)/4.0+0.5)
100 CONTINUE
   IMIN=K+1
   IMAX=K+128
   DO 200 I=IMIN,IMAX ;finish video row
      NEWUNPACK(I)=15 ;set pixels to white
   200 K=K+128 ;jump to next row
   CALL REPACK(2048,NEWUNPACK,NEWPACK)
   M2=M1/2 ;write one-half the #of blocks read
   CALL WRBLK(2,M2,NEWPACKED,8,IER)
   IF(IER.NE.1)TYPE"WRBLK ",M2," error:",IER
   DO 500 J=1,4096
      OLDPACKED(J)=177777K
500 CALL WRBLK(2,J,OLDPACKED,16,IER)
   IF(IER.NE.1)TYPE"WRBLK ",J," error:",IER
   CALL TIMER(1) ;stop timer
   WRITE(10,700)CONTFILE(1)
700 FORMAT(" The reduced picture is in the file ----> ",S13)
   CALL RESET
   STOP
   END

C**********************************************************************
C Set the rest of the picture to white (15) for the most
C aesthetic display. These pixels are later set to zero
C by the program NORMALIZE.
C
DO 500 J=1,4096
   OLDPACKED(J)=177777K
500 CALL WRBLK(2,J,OLDPACKED,16,IER)
   IF(IER.NE.1)TYPE"WRBLK ",J," error:",IER
   CALL TIMER(1) ;stop timer
   WRITE(10,700)CONTFILE(1)
700 FORMAT(" The reduced picture is in the file ----> "S13)
   CALL RESET
   STOP
   END

C**********************************************************************
C Program REDUCE **********************************************************************
APPENDIX E:

CORRELATION IMPLEMENTATION

This appendix contains the following programs:

1. NORMALIZE
2. CMULTIPLY
3. IMULTIPLY
Program NORWLIZE Written by Lt. Jim Cromer

Fortran 5

If switch 'U' is chosen:
This program normalizes the upper left quadrant of an input packed video file into twenty-four (30 column by 20 row) normalized grid blocks, with the average sum of the squares of the normalized pixel values per unit template window area equal to one (assumes a 23x47 reduced template window).

If switch 'L' is chosen:
The entire lower right quadrant is normalized to give it an energy of unity. Switch L is usually used to normalize templates.

Otherwise:
The program will ask the user to input the number of horizontal and vertical grid rectangles. Choose from:
- Horizontal -> 1,2,3,4,5,6,8,10,
- Vertical -> 1,2,3,5,6,10,15,30

The output file is a 256x256 element complex contiguous file (or random, if a contiguous file cannot be created). The program assumes the input file is a reduced picture in the upper left or lower right hand quadrants. It is normally used in the sequence:
- REDUCE NORWLIZE DIRECT CMULTIPLY INVERSE <-

Execution Line Format:
- NORMALIZE[/U or L] infile outfile
  One of the switches must be selected for use as an automatic program, as in a macro file; a U indicates to normalize the upper left-hand quadrant; an L indicates that the lower right-hand quadrant is to be normalized. Pixel values outside of the selected quadrant will be set to zero.

Load Line Format:
- NLDR NORMALIZE IOF TIMER UNPACK @FLIB@

REAL NORM(120,30), ENERGY(120,30)
INTEGER ARRAY(256), INFILE(7), OUTFILE(7), ROW(32,4,2)
INTEGER ONECOUNT, TWOCOUNT, MS(2), UNPACKED(512), MAIN(7)
INTEGER VGRID, HGRID, GRIDBLK
INTEGER HWIDTH, VNUMOFG, HNUMOFG, BPERGRID
COMPLEX CNORM(1024), CZERO
LOGICAL TEST
EQUIVALENCE (NORM, ENERGY)
ISTART=0
ISTOP=1

C
C

C***** I/O FILE MANAGEMENT ****************************
C

CALL RESET
CALL IOF(2,MAIN,INFILE,OUTFILE,IDUM,MS,I2,I3,I4)
CALL DFILW("TEMP",IER)
IF(MS(1).NE.1.AND.IER.NE.13)TYPE"DFILW error: ",IER
CALL CFILW("TEMP",2,IER)
IF(MS(1).NE.1)TYPE"DFILW error: ",IER
IER=1
JER=1
CALL OPEN(2,OUTFILE,2,IER)
IF(IER.EQ.1)GO TO 55
TYPE"Attempting to create a contiguous file."
IF(IER.EQ.13)CALL CFILW(OUTFILE,3,1024,JER)
IF(JER.NE.4)TYPESuccessfully created a contiguous file."
IF(JER.EQ.41)TYPE"Must create a random file instead."
IF(JER.EQ.41)CALL CFILW(OUTFILE,2,JER)
IF(JER.NE.1)TYPE"DFILW error: ",IER
IF(IER.EQ.13)CALL OPEN(2,OUTFILE,3,IER)
IF(IER.NE.1)TYPE"DFILW OPEN ERROR: ",IER
55
CALL OPEN(0,INFILE,1,IER)
IF(IER.NE.1)TYPE"INFILE OPEN error: ",IER STOP
CALL OPEN(0,"TEMP",2,IER)
IF(IER.NE.1)TYPE"TEMP OPEN error: ",IER
C
C

C***** DETERMINE SWITCHES AND SET VARIABLES ***************
C

IF(MS(1).EQ.16)GO TO 500 ;switch was L
IF(MS(2).EQ.2048)GO TO 1 ;switch was U
GO TO 556

5555 TYPE"Input error! <7> 0 < input <121",<15>"
556 ACCEPT"Enter horizontal # of grid rectangles
$ (1-120): ";HNUMOFG
TEST=HNUMOFG.LT.1.OR.HNUMOFG.GT.120
IF(TEST)GO TO 5555
IHOLD=MOD(120,HNUMOFG)
IF(IHOLD.NE.0)TYPE"Try again. Input must
divide evenly into 120."
IF(IHOLD.NE.0)GO TO 556

558 ACCEPT"Enter vertical # of grid rectangles
$ (1-30): ";VNUMOFG
IHOLD=MOD(30,VNUMOFG)
TEST=VNUMOFG.LT.1.OR.VNUMOFG.GT.30
IF(TEST)TYPE"Out of range! <7> <15>"
IF(TEST)GO TO 558
IF(IHOLD.NE.0)TYPE"Try again. Input must divide
evenly into 30."
IF(IHOLD.NE.0)GO TO 558
GO TO 3
TYPE "Lower right quadrant option on."
IMIN=32
IROW=2
JMIN=0
TWOCOUNT=512
KSTART=128
GO TO 2
1 TYPE "Upper left quadrant option on."
NUMOFG=4
VNUNOFG=6
3 IROW=1
IMIN=0
JMIN=512
TWOCOUNT=0
KSTART=0

C

C****** CREATE THE NORMALIZED FILE ****************************
C
2 IMAX=IMIN+31 ;limits on infile RDBLK (lower)
JMAX=JMIN+511 ;limits on outfile zeroed blocks
CALL TIMER(KSTART) ;start timer
TYPE "Creating the normalized file."
ONECOUNT=0 ;workfile WRBLK counter
SUNSO=0.0
CZERO=COMPLEX(0.0,0.0)
DO 10 J=1,1024
     CIE(J)=CZERO
10 DO 20 J=JMIN,JMAX,16 ;zero appropriate outfile rows
     CALL WRBLK(2,J,CNORMAL,16,IER)
20 IF (IER.NE.) TYPE "CNORM zero WRBLK error: ",IER
     IF (IMIN.EQ.32) GO TO 200 ;switch was L

C

C*** NORMALIZE UPPER QUADRANT ***
C
Create the unpacked workfile and determine the energy content of it. Work on 4 picture rows per loop.
C
HWIDTH=120/NUNOFG ;width of rectangle
BPERGRID=30/VNUMOFG ;RDBLKS per grid
TLENGTH=23.0 ;length of reduced template
TWIDTH=47.0 ;width " "
VLENGTH=4.0*FLOAT(BPERGRID) ;4 rows/block
TAREA=LENGTH*TWIDTH ;template area
GAREA=FLOAT(HWIDTH)*VLENGTH ;rectangle area
AREAFAC=1.0*TAREA/GAREA
DO 19 K=1,30 ;initialize energy terms
19 DO 19 J=1,120
     ENERGY(J,K)=0.0

C

C Determine energy in 30 blocks (120 rows)
C
INBLOCK=IMIN-1 ;set RDBLK counter
DO 40 GRID=1,VNUMOFG ;do 5 rows of grids
DO 39 GROBLK=1,BPERGRID        ;of 6 blocks each
      INBLOCK=INBLOCK+1            ;GROBLK counter
      CALL GROBLK(0,INBLOCK,ARRAY,1,IER)
      KK=IMIN
     DO 25 II=1,4                   ;do 4 video rows
        DO 21 J=1,2                 ;set first 8 columns
           KK=KK+1                  ;to zero
           ROW(J,II,IRW)=0           ;(noise terms)
        CONTINUE
        DO 23 J=3,32                ;arrange non-zero portion
           KK=KK+1                  ;of picture for processing
           ROW(J,II,IRW)=ARRAY(KK)
        CONTINUE
     KK=KK+32
     CONTINUE
     CALL UNPACK(128,ROW(1,1,IRW),UNPACK)
     C
     C Determine energy in 4 rows
     C
     MINCOL=9
     DO 30 HGRID=1,HNUMOFG          ;do 5 columns of grids,
        MAXCOL=MINCOL+WIDTH-1
     DO 28 KCOL=MINCOL,MIXCOL
        DO 26 JROW=0,3               ;do 4 rows
           ENERGY(HGRID,VGRID)=ENERGY(HGRID,VGRID)+
           (FLOAT(UNPACKD(KCOL+JROW*128)+1)**2
        CONTINUE
     CONTINUE
     MINCOL=MAXCOL+1
     CALL WRBLK(1,ONECOUNT,UNPACKED,2,IER)
     IF(IER.NE.1)TYPE="GROBLK ",ONECOUNT," error!",IER
     ONECOUNT=ONECOUNT+2           ;WRBLK counter
     CONTINUE                      ;do next block of grids
     CONTINUE                      ;do next grid row
     IF(HNUMOFG.GT.5)GO TO 666
     WRITE(12,3000)INFIL(1)
     3000 FORMAT("ENERGIES OF ",13,///)
     DO 100 VGRID=1,HNUMOFG
     WRITE(12,2000)VGRID,(ENERGY(HGRID,VGRID),HGRID=1,HNUMOFG
     2000 FORMAT("GRID ROW",12,5(10X,F12.2),/)  
     100 CONTINUE
     C
     C Determine the normalization factors
     C
     666 DO 50 VGRID=1,VNUMOFG       ;vertical
          DO 45 HGRID=1,HNUMOFG      ;horizontal
          IF(ENERGY(HGRID,VGRID).LE.1.0)ENERGY(HGRID,VGRID)=1.0
          NORM(HGRID,VGRID)=SORT(ENERGY(HGRID,VGRID)*AREAPARAM)
        CONTINUE
     45 CONTINUE
     50 CONTINUE
     C
     C Normalize and create the output file
     C

INBLOCK=2 ; RBBLK counter
DO 70 VGRID=1,VNUMDFG
DO 60 GRIDBLK=1,BFGRID
 INBLOCK=INBLOCK+2
 CALL RBBLK(1,INBLOCK,UNPACKED,2,IER)
 MINCOL=9 ; starting column
 ICNT=0
 DO 58 GRID=1,HNUMDFG ; do portion of 5 grid blocks
 MAXCOL=MINCOL+WIDTH-1 ; width=24
 DO 56 KCOL=MINCOL,MAXCOL ; do width of 1 grid block
 ICNT=ICNT+1
 DO 54 JROW0=1,32
 CALL WRBLK(2,TEMISON,32,IER)
 IF (IER.NE.1) TYPE "2WRBLK ", TEMISON, " error ":, IER
 TEMISON=TEMISON+32
 DO 52 J=1,2 ; zero out noise rows
 CALL WRBLK(2,TEMISON,32,IER)
 IF (IER.NE.1) TYPE "2WRBLK ", TEMISON, " error ":, IER
 TEMISON=TEMISON+32
 CONTINUE
 GO TO 555
 C
 **** Normalize lower quadrant (template) ****
 C
 DO 240 I=IMIN,IMAX ; read non-zero portion of infile
 CALL RBBLK(0,I,ARRAY,1,IER)
 IF (IER.NE.1) TYPE "0RBBLK ", I, " error ":, IER
 C
 Set nonzero portion of ARRAY equal to ROW
 C
 I=IMIN
 DO 225 II=1,4
 DO 223 J=1,32
 KK=KK+1
 JROW(J,II,IROW)=ARRAY(KK)
 223
 KK=KK+32
 225
 C
 CALL UNPACK(128,ROW(1,1,IROW),UNPACKED)
 DO 230 J=1,512 ; determine energy in 4 rows
 SUMSQ=SUNSQ+UNPACKED(J)**2
 CALL WRBLK(1,ONECOUNT,UNPACKED,2,IER)
 230

IF (IER.NE.1) TYPE "1WRBLK ", ONECOUNT, " error: ", IER
ONECOUNT=ONECOUNT+2

TENERGY=SQRT(SUMSQ) ; the normalizing factor

* Normalize the significant pixels of INFILE **************

DO 270 I=0,62,2
  KK=KSTART ; starting column of nonzero outfile
  ICNT=0
  CALL RDBLK(1,I,UNPACKED,2,IER)
  IF (IER.NE.1) TYPE "1RDBLK ", I, " error: ", IER
  DO 260 K=1,4 ; normalize infile
    DO 250 J=1,128
      KK=KK+1
      ICNT=ICNT+1
      OUTPUT=UNPACKED(ICNT)/TENERGY
      CNORM(KK)=CMPLX(OUTPUT,0.0)
  250
  260 KK=KK+128
  CALL WRBLK(2,TCOUNT,CNORM,16,IER)
  IF (IER.NE.1) TYPE "2WRBLK ", TCOUNT," error: ", IER
  TCOUNT=TCOUNT+16

C
C
C

CALL TIMER(ISTOP) ; stop timer
WRITE(10,1000)OUTFILE(1)
1000 FORMAT(" The normalized file is in --> ",S13)
CALL RESET
CALL DFILW(" TEMP",IER)
IF (IER.NE.1) TYPE "TEMP DFILW error: ", IER
STOP
END

C
C
C

 Program NORMALIZE ***********************
Program CMULTIPLY Written by Lt. Jim Cromer

Portran 5

This program performs a point-by-point complex multiplication between INFILE1 and CONJG(INFILE2). The input files must be 256x256 point complex arrays; the output file is a 256x256 point complex array.

Execution Line Format:
CMULTIPLY infile1 infile2 outfile

Load Line Format:
RLDR CMULTIPLY IOF @FLIB@

INTEGER FILE(7), INFILE1(7), INFILE2(7), MAIN(7)
COMPLEX MA1(2048), MA2(2048), MA3(2048)
COMMON FILE, INFILE1, INFILE2, MAIN

****** I/O FILE MANAGEMENT ********************

CALL IOF(3, MAIN, INFILE1, INFILE2, FILE, MS, I2, I3, I4)
WRITE(10, 1999) INFILE1(1), INFILE2(1), FILE(1)
CALL OPEN(1, FILE, 2, IER)
IF (IER .EQ. 1) GO TO 55
CALL CFILW(FILE, 3, 1024, IER)
IF (IER .EQ. 41) CALL CFILW(FILE, 2, IER)
IF (IER .NE. 1) TYPE "OUTFILE CFILW error ", IER
CALL OPEN(1, FILE, 2, IER)
IF (IER .NE. 1) TYPE "OUTFILE OPEN error ", IER
55 CALL OPEN(2, INFILE1, 2, IER)
IF (IER .NE. 1) TYPE "INFILE1 OPEN error ", IER
CALL OPEN(3, INFILE2, 2, IER)
IF (IER .NE. 1) TYPE "INFILE2 OPEN error ", IER

****** Perform point-by-point multiplication *******

DO 30 I=0,992,32
CALL RDBLK(2, I, MA2, 32, IER) ; read 8 complex rows
IF (IER .NE. 1) TYPE "RDBLK ", I, " error: ", IER
CALL RDBLK(3, I, MA3, 32, IER)
IF (IER .NE. 1) TYPE "RDBLK ", I, " error: ", IER
DO 20 K=1,2048
MA1(K) = MA2(K) * CONJG(MA3(K))
20 CONTINUE
CALL WRBLK(1, I, MA1, 32, IER)
IF (IER .NE. 1) TYPE "WRBLK ", I, " error: ", IER
DO 30 K=1,2048
MA1(K) = MA2(K) * CONJG(MA3(K))
30 CONTINUE
WRITE(10, 40) FILE(1)
FORMAT(* " , S13," created by CMULTIPLY")

END

******** Program CMULTIPLY **************************
Program CTOI Written by Lt. Jim Cromer

Fortran 5

This program converts a 256x256 complex file into a 256x256 integer file. The real part only of the complex file is saved; the imaginary part is assumed to be zero. The maximum value and its position are written to first 3 words of block 255. [Values greater than 2 are set to 1. All other values are divided by two.]

Execution Line Format:
CTOI complex infile integer outfile

Load Line Format:
RLDR CTOI IOF TIMER @LIB@

C** I/O FILE MANAGEMENT ****************************************************

CALL IOF (2,MAIN,INFILE,OFILE,IDUM,MS,IS1,IS2,IS3)
CALL TIMER (0) ; start timer
CALL OPEN (0,INFILE,1,IER)
IF (IER .NE.) TYPE "INFILE OPEN error ",IER
CALL OPEN (1,OFILE,3,IER)
IF (IER .EQ. 1) GO TO 1
CALL CFILW (OFILE,3,256,IER)
IF (IER .EQ. 1) TYPE "Created a contiguous output file."
IF (IER .EQ. 41) CALL CFILW (OFILE,2,IER)
IF (IER .NE. 1) TYPE "OFILE CFILW error ",IER
CALL OPEN (1,OFILE,3,IER)
IF (IER .NE. 1) TYPE "OFILE error ",IER

C** CONVERT COMPLEX WORDS TO INTEGER **************************************

IMAX=0
DO 20 J=0,960,64
CALL RDMLK (0,J,INCOMPLEX,64,IER)
IF (IER .NE. 1) TYPE "INCOMPLEX RDMLK ",J," error: ",IER
DO 10 K=1,4096
AREAL=REAL(INCOMPLEX(K))/2.0
OUTINTEGER (K)=INT (AREAL*32767.0)
IF (AREAL.GE.1.0) OUTINTEGER(K)=32767
IF (OUTINTEGER(K).LT.IMAX) GO TO 10
COLUMNNUMBER=K-(INT ((K-1)/256))*6
ROWNUMBER=INT ((K-1)/256)+1+J/4
IMAX=OUTINTEGER(K)
10 CONTINUE
17 CALL WRBLK (1,(J/4),OUTINTEGER,16,IER)
IF(IER.NE.1)TYPE"OUTINTEGER WRBLK ",
$(J/2)," error ",IER
20 CONTINUE
OUTINTEGER(1)=IMAX
OUTINTEGER(2)=COLUMNNUMBER
OUTINTEGER(3)=ROWNUMBER
CALL WRBLK(1,255,OUTINTEGER,1,IER)
IF(IER.NE.1)TYPE"IMAX WRBLK error: ",IER
CALL TIMER(1)
;stop timer
TYPE" The maximum integer value is: ",IMAX
TYPE": @ ROWNUMBER=",ROWNUMBER
TYPE": @ COLUMN =",COLUMNNUMBER
RMAX=(FLOAT(IMAX))/32767.0
TYPE": IMAX/32767 =",RMAX
WRITE(10,1000)INFILE(1),OUTFILE(1)
1000 FORMAT(" The complex file ",S8," has been
$converted to the integer file ",S8)
CALL RESET
STOP
END

C********** Program CTOI ******************************************
Program IMULIFLY Written by Lt. Jim Cromner

This program calculates the point-by-point geometric mean between INFILE1 and INFILE2; the product is output to OUTFILE. All files must be 256x256 integer files.

Execution Line Format:
IMULIFLY infile1 infile2 outfile

Load Line Format:
RLDR IMULIFLY IOF @FLIB@

INTEGER MAIN(7), INFILE1(7), INFILE2(7), OUTFILE(7)
INTEGER GS(2), LS1(2), LS2(2), LS3(2)
INTEGER FACTOR1(8192), FACTOR2(8192), PRODUCT(8192)

**************** I/O FILE MANAGEMENT ***************

CALL IOF(3, MAIN, INFILE1, INFILE2, OUTFILE, GS, LS1, LS2, LS3)
JER=1
KER=1
CALL OPEN(1, INFILE1, 2, IER)
IF (IER .NE. 1) STOP " INFILE1 OPEN ERROR"
CALL OPEN(2, INFILE2, 2, IER)
IF (IER .NE. 1) STOP " INFILE2 OPEN ERROR"

First check to see if the file exists; if it doesn't, try to create a contiguous file.

CALL OPEN(3, OUTFILE, 2, IER)
LER=IER
IF (IER .EQ. 13) CALL CFILW (OUTFILE, 3, 256, JER)
IF (IER .NE. 1) CALL CFILW (OUTFILE, 2, KER)
IF (IER .NE. 1) STOP " OUTFILE CFILW ERROR"
IF (KER .EQ. 1 .OR. JER .EQ. 1) IER=1
IF (IER .NE. 1) STOP " OUTFILE OPEN ERROR"
IF (IER .NE. 1) CALL OPEN(3, OUTFILE, 2, IER)
IF (IER .NE. 1) STOP " OUTFILE OPEN ERROR"

******** Perform point-by-point multiplication ********

DO 20 I=0,224,32
CALL RDBLK(1, I, FACTOR1, 32, IER)
IF (IER .NE. 1) TYPE " RDBLK #", I, " error:", IER
CALL RDBLK(2, I, FACTOR2, 32, IER)
IF (IER .NE. 1) TYPE " RDBLK #", I, " error:", IER
DO 10 J=1,8192
   AHOOLD=SQRT((FLOAT (FACTOR1(J)) * FLOAT (FACTOR2(J))))
   PRODUCT(J)=INT (AHOOLD + 0.5)
10   CONTINUE
   CALL WRBLK(3, I, PRODUCT, 32, IER)
20   CONTINUE
IF (IER .NE. 1) TYPE "WRBLK "$, I," error: ", IER
  TYPE* *** DATA CRUNCH! ***<11><11>*** DATA CRUNCH! ***
  CONTINUE
C
C****** Write message to the CRT terminal ********
C
  WRITE(10,1000)OUTFILE(1)
  1000 FORMAT('/','The integer product file is named ----> ",Sl3,/
  CALL RESET
  STOP
  END
C
C******* Program IMULTIPLY **********************************************
APPENDIX F:
PROCESS EVALUATION

This appendix contains the following programs:

1. ITOC
2. PEAK (F1)
3. CTOW
4. DISTANCE (EUCLID)
Program ITOC

Fortran 5

This program converts a 256x256 integer file into a 256x256 complex file.

Execution Line Format:

ITOC/[A,E,H,N, or O] integerfile[/C and or T] complex[/M]

Global switches —> A: convert all values >0
>values not converted will be set to zero

C:
>convert values >80% of the maximum

E: convert values >1/2 maximum

H: convert values >90% maximum

N: convert values >90% maximum

O: accept other conversion value

Inputfile switch —> C: "crunch" 256x256 data into 256x128 array for use with PLTRINS contour

T: set output values equal to input value minus the threshold

Outputfile switch —> M: insert maximum value into 3rd column of every 4th row

Load Line Format:

RLDR ITOC IOF TIMER @FLIB@

***************************************************************

INTEGER MAIN(7),INFILE(7),OUTFILE(7),MS(2),IS2(2),IS1(2)
INTEGER ININTEGER(4096),COLUMN,ROW
REAL INREAL

C***************************************************************

CALL IOF(2,MAIN,INFILE,OUTFILE,IDUM,MS,IS1,IS2,IS3)
PERCENT=9999.0
IF(ITEST(MS(1),15),EQ.1)PERCENT=0.0 ; switch was A
IF(ITEST(MS(1),11),EQ.1)PERCENT=0.80 ; switch was E
IF(ITEST(MS(1),8),EQ.1)PERCENT=0.50 ; switch was H
IF(ITEST(MS(1),2),EQ.1)PERCENT=0.90 ; switch was N
2 IF(ITEST(MS(1),1),EQ.1)ACCEPT" Enter the $ threshold percent (0.0 - .99): ",PERCENT
IF(PERCENT.LT.0.0.OR.PERCENT.GT.1.0)GO TO 2
IF(PERCENT.EQ.-9999.0)STOP"BAD GLOBAL SWITCH"

CALL TIMER(0) ; start timer
CALL OPEN(0,INFILE,1,IER)
IF(IER.NE.1)TYPE"INFILE OPEN error ",IER
CALL OPEN(1,OUTFILE,3,IER)
IF(IER.NE.1)GO TO 1
CALL CFILW(OUTFILE,3,1024,IER)
IF(IER.NE.1)TYPE"Created a contiguous output file."
IF(IER.NE.1)CALL CFILW(OUTFILE,2,IER)
IF(IER.NE.1)TYPE"OUTFILE CFILW error ",IER

127
CALL OPEN(1,OUTFILE,3,IER)
IF(IER.NE.1)TYPE"OUTFILE error ",IER

C********* CONVERT INTEGER WORDS TO COMPLEX ************

1 CALL RDKLK(0,255,INTEGRER,1,IER)
IF(IER.NE.1)TYPE"INTEGRER RDLK ERROR: ",IER
IMAX=INTEGRER(1)
COLUMN=INTEGRER(2)
RT=INTEGRER(3)
ITHRESH=INT(PERCENT*FLOAT(IMAX))
JFACTOR=2
JBLOCK=64
INCREMENT=0
IF(IS1(1).EQ.8192)JFACTOR=1
IF(IS1(1).EQ.8192)JBLOCK=32
IF(IS1(1).EQ.8192)INCREMENT=256

Create the output file

DO 20 J=0,480,32
 CALL RDLK(0,(J/2),INTEGRER,16,IER)
IF(IER.NE.1)TYPE"INTEGRER RDLK ",,(J/2)," error:",IER
LL=0
KK=0
DO 10 I=1,(8*JFACTOR)
    KK=KK+1
    LL=LL+1
    IF(IS1(2).NE.4096)GO TO 3
    INREAL=FLOAT(INTEGRER(KK)-ITHRESH)/32767.0
    GO TO 4
    INREAL=FLOAT(INTEGRER(KK))/32767.0
4 IF(INTEGRER(KK).LT.IITHRESH)INREAL=0.0
    OUTCOMPLEX(LL)=CMPLX(INREAL,0.0)
    CONTINUE
7 KK=KK+INCREMENT
    CONTINUE
20 CONTINUE

Prepare output for PLTNRN row plot

IF(IS2(1).EQ.8.AND.IS1(2).EQ.4096)OUTCOMPLEX(3)=CMPLX(FLOAT(IMAX-ITHRESH),0.0)
IF(IS2(1).EQ.8.AND.IS1(2).NE.4096)
    OUTCOMPLEX(3)=CMPLX(FLOAT(IMAX),0.0)
    CALL WDLK(1,(J*JFACTOR),OUTCOMPLEX,JBLOCK,IER)
    IF(IER.NE.1)TYPE"OUTCOMPLEX WDLK ",,(J*JFACTOR),
    "$ error ",IER
    CONTINUE
20 CONTINUE

The data will be compressed to the front half of the output plane.

IF(IS1(1).NE.8192)GO TO 35
DO 23 I=1,3
     OUTCOMPLEX(I+1792)=CMPLX(0.0,0.0)
23    CONTINUE
     CALL WRBLK(1,480,OUTCOMPLEX,32,IER)
     IF(IER.NE.1)TYPE"WRBLK (DATA) #480 error:" ,IER
     DO 25 I=1,4096
         OUTCOMPLEX(I)=CMPLX(0.0,0.0)
25    CONTINUE
     CALL WWLK(1,480,WRUNHX,32,IER)
     IF(IER.NE.1)TYPE"WRBLK (DATA) #480 error:" ,IER
     DO 30 J=512,960,64
         CALL WRBLK(1,J,OUTCOMPLEX,64,IER)
         IF(IER.NE.1)TYPE"WRBLK #" ,J," error:" ,IER
30    CONTINUE
     GO TO 40

C Expanded output only
C
C DO 38 I=1,3
     OUTCOMPLEX(I+3840)=CMPLX(0.0,0.0)
38    CONTINUE
     CALL WRBLK(1,960,OUTCOMPLEX,64,IER)
     IF(IER.NE.1)TYPE"WRBLK (DATA) #960 error:" ,IER

C Send completion message to CRT
C
C CALL TIMER(1);stop timer
     TYPE"IMAX=" ,IMAX
     TYPE"ITHRESH=" ,ITHRESH
     TYPE"PERCENT=" ,PERCENT
     TEMP=FLOAT(ITHRESH)/32767.0
     TYPE"Normalized threshold=" ,TEMP
     WRITE(10,1000)INFILE(1),OUTFILE(1)
     FORMAT("The integer file ",S8," has been
$converted to the complex file ",S8)
     CALL RESET
     STOP
     END

C******** Program ITOC ******************************************
Program PEAK

Fortran 5

This program searches a 256x256 integer file for isolated regions of which all values are above a given threshold. If the input array is thought of as a 3-dimensional surface, then these regions will be the "peaks" of the surface. The position and value of both local and global peaks is determined.

Execution Line Format:

PEAK

Load Line Format:

RLDR PEAK F1 @FLIB@

Flag Values:

PEAK -> -1 - peak closed
;holds condition 0 - peak unused
;of global peaks 1 - peak open

STATUS -> -1 - row peak matched
;holds condition 0 - unused in current row
;of row peaks 1 - row peak unmatched

INSIDE -> .TRUE. - previous value
;determines if tested > threshold
;pointer in .FALSE. - else
;interior or exterior
;of a row peak

REAL ND4ALIZE (10)
INTEGER INFLE(7),WIDTH(10),PCENTMAX(10),LENGTH(10)
INTEGER VALUE(256),F1,THRESH,ISTART(10),ISTOP(10)
INTEGER ROW,COLUMN,PVALUE,TEMPPK,PMO(10),IMAX(10)
INTEGER MAXCOLUMN(10),FSTOP(10,256),FSTART(10,256)
INTEGER PVALUE(10),PCOLUMN(10),STATUS(10),PEAK(10)
INTEGER JMAX(10),JMIN(10),RANK(10),IRANK(0:10)
LOGICAL ATES7,INSIDE

C********** INITIALIZE VARIABLES ****************************

C

IOPT=0

100 DO 200 I=1,10
   IRANK(I)=0
   RANK(I)=0
   PEAK(I)=0
   PVALUE(I)=0
   PCOLUMN(I)=0
   PMO(I)=0
   DO 150 J=1,256
PSTART(I,J)=257
PSTOP(I,J)=0

150 CONTINUE
200 CONTINUE

C******** SET I/O PARAMETERS ****************************

C IF(IOPT.EQ.2)GO TO 250
ACCEP"What is the name of the input integer file? "
READ(11,1000)INFILE(1)
1000 FORMAT(S13)
CALL OPEN(0,INFILE,2,IER)
IF(IER.NE.1)STOP"INFILE OPEN ERROR"
CALL RDBLK(0,255,VALUE,1,IER)
MAXIMUM=VALUE(1)
TYPE"Absolute max="MAXIMUM
GO TO 250
210 TYPE"Input error. Percentage must be between 1-100.<7><15>"
250 ACCEPT"Enter integer percentage of absolute maximum
$ to be included: ",I PERCENT
IF(I PERCENT.LT.1.OR.I PERCENT.GT.100)GO TO 210
THRESH=INT(FLOAT(I PERCENT)*FLOAT(MAXIMUM)/100.0)
TYPE"INTEGER THRESHOLD="THRESH

C******** Loop through the scan and matching modules 254 times
C (test all but first and last rows)

C DO 600 ROW=-2,255 ;test rows 2-255
CALL RDBLK(0,(ROW-1),VALUE,1,IER)
IF(IER.NE.1)TYPE"RDBLK ", (ROW-1)," error: ",IER
INSIDE=.FALSE.
NUMPEAKS=0
DO 300 I=1,10
STATUS(I)=0
300 CONTINUE

C****** SCAN ROW ****************************

C This module determines if the scanning pointer is in
C the interior or the exterior of a row peak. When a
C row peak is encountered, the peak counter NUMPEAKS is
C increased, and the flag STATUS is set to the unmatched
C condition. The maximum value and corresponding column
C number for each row peak is stored.

C DO 350 COLUMN=2,255
FVALUE=FL1(VALUE(COLUMN-1),VALUE(COLUMN),VALUE(COLUMN+1))
IF(FVALUE.LT.THRESH.AND.,NOT.INSIDE)GO TO 350
IF(FVALUE.GE.THRESH.AND.INSIDE)GO TO 320
IF(FVALUE.GE.THRESH.AND.,NOT.INSIDE)GO TO 310
IF(FVALUE.LT.THRESH.AND.INSIDE)GO TO 330
310 NUMPEAKS=NUMPEAKS+1 ; row-peak counter
STATUS (NUMPEAKS) = 1 ; row peak opened, unmatched
IMAX (NUMPEAKS) = 0
ISTOP (NUMPEAKS) = 256
ISTART (NUMPEAKS) = COLUMN
INSIDE = .TRUE.
320 IF (IMAX (NUMPEAKS) .GT. FVALUE) GO TO 350
    IMAX (NUMPEAKS) = FVALUE
    MAXCOLUMN (NUMPEAKS) = COLUMN
    GO TO 350
330 INSIDE = .FALSE.
    ISTOP (NUMPEAKS) = COLUMN-1
350 CONTINUE
C
C
C
C
If no values above threshold were found in the
last row tested, then close all open peaks.
C
C
IF (NUMPEAKS .NE. 0) GO TO 400
C
DO 370 I=1,10
    IF (PEAK (I) .NE. 1) GO TO 370
    PEAK (I) = 1
    JMAX (I) = ROW-1
370 CONTINUE
C
GO TO 600
C
400 CONTINUE
C
C
C******** ATTEMPT TO MATCH **************************************
C
Attempt to match row peaks to open global peaks.
C
DO 500 I=1,10
    IF (PEAK (I) .NE. 1) GO TO 500 ; check all open peaks
    PEAK (I) = 1 ; will be closed unless matched
    JMAX (I) = ROW-1
    DO 450 TEMPPEAK = 1, NUMPEAKS
        IF (STATUS (TEMPPEAK) .EQ. 0) GO TO 450
        ATTEST = PSTOP (I, (ROW-1)) .LT. ISTART (TEMPPEAK)
        .OR. PSTART (I, (ROW-1)) .GT. ISTOP (TEMPPEAK)
        IF (ATTEST) GO TO 450 ; did not match
        PEAK (I) = 1
        STATUS (TEMPPEAK) = 1
        PSTART (I, ROW) = MIN (PSTART (I, ROW), ISTART (TEMPPEAK))
        PSTOP (I, ROW) = MAX (PSTOP (I, ROW), ISTOP (TEMPPEAK))
        IF (FVALUE (I) .GE. IMAX (TEMPPEAK)) GO TO 450
        FVALUE (I) = IMAX (TEMPPEAK)
        PRW (I) = ROW
        PCOLUMN (I) = MAXCOLUMN (TEMPPEAK)
450 CONTINUE
C
500 CONTINUE
C
132
C******** MUST OPEN A NEW GLOBAL PEAK ****************************
C
   Match unmatched row peaks to unused global peaks.
C
   DO 550 TEMPPEAK=1,NUMPEAKS
      IF (STATUS(TEMPPEAK).NE.1) GO TO 550
      DO 510 I=1,10
          IF (STATUS(TEMPPEAK).NE.1) GO TO 510
          IF (PEAK(I).NE.0) GO TO 510
          STATUS(TEMPPEAK)=1
          PEAK(I)=1
          JMIN(I)=ROW
          JMAX(I)=ROW
          PSTOP(I,ROW)=ISTOP(TEMPPEAK)
          PSTART(I,ROW)=ISTART(TEMPPEAK)
          PVALUE(I)=IMAX(TEMPPEAK)
          PROW(I)=ROW
          PCOLUMN(I)=MAXCOLUMN(TEMPPEAK)
          TYPE="PEAK ",I," START ROW: ",ROW
   510  CONTINUE
   550  CONTINUE
   600  CONTINUE
C
C******** ELIMINATE MULTIPLE DEFINED PEAKS ***********************
C
   DO 650 I=1,9
      IF (PEAK(I).EQ.0) GO TO 650
      DO 620 J=1,10
          IF (PEAK(J).EQ.0) GO TO 620
          IF (I.EQ.J) GO TO 620
          IF (PVALUE(J).NE.PVALUE(I)) GO TO 620
          IF (PROW(J).NE.PROW(I)) GO TO 620
          IF (PCOLUMN(J).NE.PCOLUMN(I)) GO TO 620
          TYPE="*** MULTIPLE DEFINED PEAK FOUND ***"
          PEAK(J)=999
          JMIN(I)=MIN(JMIN(I),JMIN(J))
          JMAX(I)=MAX(JMAX(I),JMAX(J))
          JSTART=JMIN(I)
          JSTOP=JMAX(I)
          DO 610 ROW=JSTART,JSTOP
              PSTOP(I,ROW)=MAX(PSTOP(I,ROW),PSTOP(J,ROW))
              PSTART(I,ROW)=MIN(PSTART(I,ROW),PSTART(J,ROW))
       610  CONTINUE
       620  CONTINUE
   650  CONTINUE
   700  CONTINUE
C
C******** SORT PEAKS ACCORDING TO THEIR MAXIMUM VALUES ***********
C
   DO 706 K=1,10
      LOCOUNT=0
      DO 704 I=1,10
          IF (PEAK(I).EQ.999) PVALUE(I)=0
       704  CONTINUE
  706  CONTINUE
IF(PEAK(I).EQ.0)GO TO 704
IF(PVALUE(I).EQ.32767)PVALUE(I)=32766
ICOUNT=0
DO 702 J=1,10
   IF(PEAK(J).EQ.0)GO TO 702
   IF(I.EQ.J)GO TO 702
   ATEST=PVALUE(I).EQ.PVALUE(J)
   IF(.NOT.ATEST)GO TO 702
   LCOUNT=ICOUNT+1
   ICOUNT=ICOUNT+1
   PVALUE(J)=PVALUE(J)-ICOUNT
702 CONTINUE
704 CONTINUE
   IF(ICOUNT.EQ.0)GO TO 708
706 CONTINUE
708 IRANK(0)=32767
DO 720 I=1,10
   IF(PEAK(I).EQ.0)GO TO 720
   DO 710 J=1,10
      IF(PEAK(J).EQ.0)GO TO 710
      IF(PVALUE(J).GE.IRANK(I-1))GO TO 710
      IRANK(I)=MAX(IRANK(I),PVALUE(J))
    710 CONTINUE
720 CONTINUE
DO 800 I=1,10
   IF(PEAK(I).EQ.0)GO TO 800
   DO 750 J=1,10
      IF(IRANK(J).EQ.0)GO TO 750
      IF(PVALUE(I).EQ.IRANK(J))RANK(I)=J
750 CONTINUE
800 CONTINUE
C
C******** COMPUTE OUTPUT VALUES **********************
C
810 DO 830 I=1,10
   IF(PEAK(I).EQ.999)PEAK(I)=0
   IF(PEAK(I).EQ.0)GO TO 830
   PCEMTMAX(I)=INT(PVALUE(I)/(0.01*MAXIMUM))+0.5
   NORMALIZE(I)=FLOAT(PVALUE(I))/32767.0
   LENGTH(I)=JMAX(I)-JMIN(I)+1
   WIDTH(I)=0
   JSTART=JMIN(I)
   JSTOP=JMAX(I)
   DO 820 J=JSTART,JSTOP
      IHOLD=PSTRT(I,J)-PSTRT(I,J)+1
      WIDTH(I)=MAX(IHOLD,WIDTH(I))
820 CONTINUE
830 CONTINUE
850 ICH=10
870 IF(ICH.EQ.12)WRITE(12,1500)
1500 FORMAT(/),/)
WRITE(ICH,2000)
   IF(ICH.EQ.10)WRITE(10,3000)INF0LE(I)
   IF(ICH.EQ.12)WRITE(12,4000)INF0LE(I)
RTHRESH=FLOAT(TRESH)/32767.0
WRITE(ICH,5000)RTHRESH,IPERCENT
WRITE(ICH,6000)
WRITE(ICH,7000)
WRITE(ICH,8000)

2000 FORMAT(/,'15X,"*************************************************************
                "","S13,"
3000 FORMAT(/,'30X,"INTEGER FILE EVALUATED --> <10>"",S13,"
4000 FORMAT(40X,"THRESHOLD="F5.3,"/36X,"% OF MAX PEAK:"",I5,"/)
5000 FORMAT(21X,"PEAK %MAX",33X,"NORMALIZED")
6000 FORMAT(21X,"# PEAK ROW COLUMN WIDTH LENGTH
                "$VALUE")
8000 FORMAT(21X,"--- --- --- --- --- --- --- --- ---
                "$---

   DO 910 I=1,10
   DO 900 J=1,10
       IF(RANK(J).NE.I)GO TO 900
       WRITE(I1CH,9000)I,PCH(MAX(J),PRAW(J)/PCOLUMN(J),WIDTH(J)
       WIDTH(J),NORMALIZE(J)
   9000 FORMAT(15X,/I9,I7,I7,I8,I7,I7,F12.3)
   C 9000 FORMAT(19X,I5,2X,I5,2X,I5,2X,I5,1X,I6,2X,I5,1X,F11.3)
   900 CONTINUE
   910 CONTINUE
   WRITE(I1CH,2000)
   IF(ICH.EQ.12)GO TO 950
   ACCEPT"Enter a 1 to send results to the lineprinter: ",I
   IF(I.NE.1)GO TO 950
   I=ICH
   CH=12
   GO TO 870
   950 TYPE<15>"","What next?"
       TYPE<15>"","Here are the options:<15>"
       TYPE<15>"","<11>1 — Try a new input file"
       TYPE<15>"","<11>2 — Try another threshold value"
       TYPE<15>"","<11>3 — STOP<15>"
   970 ACCEPT"Enter option --> ",IOPT
       I=IOPT
       IF(I.EQ.1.OR.I.GT.3)TYPE<15>"","Input error<7><7>1<15>"
       IF(I.EQ.1.OR.I.GT.3)GO TO 970
       IF(I.EQ.2)GO TO 100
       CALL RESET
       IF(I.EQ.1)GO TO 100
       TYPE<15>"","*** EXITING PROGRAM PEAK ***<15>"
   STOP
END

C
C********** Program PEAK ************

135
INTEGER FUNCTION F1(NBEFORE,N,NAPTER)
C******************************************************************************
C
C      Function F1
C
C      This function is part of the program PEAK. In PEAK, a function of N F1(N) is compared to a threshold of some % of the maximum value to determine if a local peak was found.

C******************************************************************************

F1=N
C
Other possible functions:
C
F1=INT(FLOAT(NBEFORE+N+NAPTER)/3.0+0.5)
F1=INT(FLOAT(NBEFORE+2*N+NAPTER)/4.0+0.5)
F1=INT(FLOAT(NBEFORE+3*N+NAPTER)/5.0+0.5)
RETURN
END
C
C****** Function F1 ***********************************************
Program CTOV by Lt Jim Cromer
Fortran 5

This program converts a complex input file (imaginary part assumed zero) into a video output file. The input file is linearly scaled to a 0-15 output range. Minimum and maximum values to be included are input by the user.

Execution Line Format:
CTOV

Load Line Format:
RLDR CTOV XWBBLK TIMER @LIB@

REAL LOWER
INTEGER IARRAY(1024),CINFILE(7),VOUTFILE(7)
COMPLEX CARRAY(1024)

C******* I/O FILE MANAGEMENT ***************
C
ACCEPT"Enter name of complex input file: 
READ(11,2000)CINFILE(1)
2000 FORMAT(S13)
ACCEPT"Enter name of video output file: 
READ(11,2000)VOUTFILE(1)
CALL TIMER(0)
CALL DFILW(VOUTFILE,IER)
IF(IER.NE.1.AND.IER.NE.13)STOP"DFILW ERROR"
CALL CFILW(VOUTFILE,2,IER)
IF(IER.NE.1)STOP"CFILW ERROR"
CALL OPEN(1,CINFILE,2,IER)
IF(IER.NE.1)STOP"OPEN ERROR"
CALL OPEN(2,VOUTFILE,2,IER)
IF(IER.NE.1)STOP"OPEN ERROR"

C******* Determine maximum and minimum values **********
C
RMAX=0.0
RMIN=99999.99
DO 2 K=0,63
   CALL RD(ILK(1,(K*16)),CARRAY,16,IER)
   IF(IER.NE.1)TYPE"1 RDILK #",(K*16)," error: ",IER
   DO 1 J=1,1024
      A=REAL(CARRAY(J))
      RMAX=AMAX1(RMAX,A)
      RMIN=AMIN1(RMIN,A)
   1 CONTINUE
   IF(MOD((K+1),4).EQ.0)TYPE"BLOCK",(K*16)," searched."
     CONTINUE
C
C******* Determine the linear scale to be used ***********
C

TYPE"Maximum=",RMAX," Minimum=" ,RMIN
ACCEPT" Enter maximum to be included: " ,UPPER
ACCEPT" Enter minimum to be included: " ,LOWER
SCALE=15.999/(UPPER-LOWER)

C
C********* Create the output file ***************
C
DO 20 K=0,63
   CALL RDMLK(1,(K*16),CARRAY,16,IER)
   IF (IER.NE.1) TYPE" 1 RDMLK ",((K*16)," error: ",IER
   DO 10 J=1,1024
      IARRAY(J)=15
      A=REAL(CARRAY(J))
      IF (A.LT.LOWER) IARRAY(J)=0
      IF (A.GE.LOWER.AND.A.LE.UPPER)
      $ IARRAY(J)=INT((A-LOWER)*SCALE)
   10 CONTINUE
   CALL WRMLK(2,K,IARRAY,1,IER)
   IF (IER.NE.1) TYPE" 2 WRMLK ",K," error: ",IER
20 CONTINUE
C
C********* Send completion message to CRT **********
C
CALL TIMER(1)
WRITE (10,1000)VOUTFILE(1)
1000 FORMAT(" The video file created is called ",SL3)
CALL RESET
STOP
END
C
C********* Program CTOV *******************************************************
Program DISTANCE Written by Lt. Jim Cromer
Fortran 5 16 Oct 1982

This program accepts as input a template file window and up to 10 local correlation peak positions found by the program PEAK. Three distance factors will be calculated between the template window and 9 scene windows (the center window corresponds to the input correlation peak, the other 8 are its nearest neighbors). The score is computed as the cube root of the product of the factors. If any factor is less than zero (corresponding to the measure for a constant gray level input scene) it is set to zero. Results are output to the lineprinter.

FACTORS USED:

\[
\begin{align*}
L1FACT &= 100(1 - N1L/NL1MAX) \\
L2FACT &= 100(1 - N2L/NL2MAX) \\
CFACT &= 100(NXY - NXYMIN)/(1 - NXYMIN)
\end{align*}
\]

where 
\(N1L\) is the normalized \(L1\) distance \\
\(N2L\) is the normalized \(L2\) distance \\
\(NXY\) is the cross-correlation value \\
\(NL1MAX\) is the \(NL1\) for a constant gray level input \\
\(NL2MAX\) is the \(NL2\) for a constant gray level input \\
\(NXYMIN\) is the \(NXY\) for a constant gray level input

Execution Line Format:
DISTANCE

Load Line Format:
RLDR DISTANCE TEST XRDBLK EUCLID @FLIB@

REAL NL1MAX
REAL NL1DIST,L1DIST,NSUMSQ,NDIST(10),NDISTL1(100)
REAL CORPEAK(100),DISTL1(10),NXYMIN,NL2MAX,NL2
INTEGER INFIL1(7),INFIL2(7),WIDTH,TB,COMMENT(200),
$SB1, SLS, TTOP, TBL, SB, CFACT(100), L2FACTOR(100), L1FACTOR(100),$
$SLSL, TLEFT, SCENE(1024), TEMP(1024), SLEFT(100), POLYMNN(10),$
$SCOLENT(100), ROWCENT(100), DIST(10), CTOP(100), PROW(10), SCOE$
LOGICAL REDUCED, LTEST, SUPPRESS

COMMON /LIST1/ SCENE, TEMP, SLS, TLS, L1DIST, NL1DIST
COMMON /LIST2/ LENGTH, WIDTH, IROWCOUNT, SUMSQ, NSUMSQ
COMMON /LIST3/ SL1ENERGY, TL1ENERGY, SENERGY, TENERGY
COMMON /LIST4/ S1NORM, S2NORM, T1NORM, T2NORM, CORREL
LTEST=.TRUE.

NL1MAX=0.46 ; normalized \(L1\) distance between the template and a constant gray level
C
99 ACCEPT"Enter template file name: "
READ(11,1000)INFILE1(1)
1000 FORMAT(S13)
   ACCEPT"Enter scene file name:"
   READ(11,1000)INFILE2(1)
   CALL OPEN(1,INFILE1,1,IER)
   IF(IER.NE.1)TYPE"INFILE1 OPEN ERROR ":,IER
   CALL OPEN(2,INFILE2,1,IER)
   IF(IER.NE.1)TYPE"INFILE2 OPEN ERROR ":,IER

C
C******** ENTER WINDOW PARAMETERS **********************
C
The choice is given to compare the original 256x256
pictures, or the reduced 128x128 versions. Reduced scene
files are assumed to occupy the upper left quadrant, tem-
plates are assumed to occupy the lower right quadrant.
C
REDCED=.FALSE.
1 ACCEPT"<15> Enter a 1 to compare original video,
  $<15>$ or a 2 to compare reduced video: "I
  IF(I.LT.1.OR.I.GT.2)TYPE"<7> INPUT ERROR! <15>"
  IF(I.LT.1.OR.I.GT.2)GO TO 1
  IF(I.EQ.2)REDUCED=.TRUE.
  IF(LTEST)GO TO 5
3 ACCEPT"Enter a 1 to change template window parameters: ",IOPT
  IF(IOPT.NE.1)GO TO 10
5 ACCEPT"<15>" Enter top row of original template
  $ window (1-256): ",TTOP
  ACCEPT" Enter left column of original template
  $ window (1-256): ",TLEFT
  ACCEPT" Enter width of window (1-256): ",WIDTH
  ACCEPT" Enter length of window (1-256): ",LENGTH
  LTEST=.FALSE.
  MTOP=TTOP
  MLFT=TLEFT
  MWIDTH=WIDTH
  MLENGTH=LENGTH
  GO TO 10
9 TYPE"SORRY<7>. Number of peaks can be 1-10 only." 
10 ACCEPT"<15>" Enter # of candidate peaks: ",NUMPEAKS
  IF(NUMPEAKS.GT.10.OR.NUMPEAKS.LT.1)GO TO 9
  TTOP=MTOP
  TLEFT=MLFT
  WIDTH=MWIDTH
  LENGTH=MLENGTH
  DO 20 II=1,NUMPEAKS
   TYPE"<15>" ",<15>" ",******** PEAK" ,II," *************
   GO TO 15
13 TYPE"<15>" "Sorry. Peak row must be 1-256."
15 ACCEPT"<15>" Enter peak row number: ",PROW(II)
  IF(PROW(II).LT.1.OR.PROW(II).GT.256)GO TO 13
CTOP(II)=256+TTOP-2*PROW(II)
IF(REDUCTED)CTOP(II)=128+INT((TTOP+1)/2)-PROW(II)
GO TO 19

17 TYPE"<7>Pear column must be 1-256. Try again."
19 ACCEPT"<15>"."Enter peak column number: ",PCOLUMN(II)
IF(PCOLUMN(II).LT.1.OR.PCOLUMN(II).GT.256)GO TO 17
SLEFT(II)=256+TLEFT-2*PCOLUMN(II)
IF(REDUCTED)SLEFT(II)=128+INT((TLEFT+1)/2)-PCOLUMN(II)
ROWCENT(II)=CTOP(II)+INT(LENGTH/2)
COLCENT(II)=SLEFT(II)+INT(WIDTH/2)
IF(REDUCTED)ROWCENT(II)=CTOP(II)+INT(LENGTH/4)
IF(REDUCTED)COLCENT(II)=SLEFT(II)+INT(WIDTH/4)

20 CONTINUE
N=-1
IF(REDUCTED)LENGTH=INT((LENGTH+1)/2)
IF(REDUCTED)WIDTH=INT((WIDTH+1)/2)
IF(REDUCTED)TTOP=INT((TTOP+1)/2)+128
IF(REDUCTED)TLEFT=INT((TLEFT+1)/2)+128

ILEFT=TLEFT
ITOP=TTOP
IWIDTH=WIDTH
ILENGTH=LENGTH
SUPPRESS=.FALSE.
ACCEPT"Enter a 1 to suppress window messages: ",I2
IF(I2.EQ.1)SUPPRESS=.TRUE.
TYPE"Executing . . . . ."

C*******************************************************************************
C
C This section computes the RDBLK and EUCLID search
C window parameters, then computes the distance measures
C for each of the windows entered.
C
II=10
DO 600 JJ=1, NUMPEAKS
DO 600 J=1,3 ;DO 9 windows
DO 600 K=1,3 ;compute the window shift
II=II+1

COLENT(II)=COLENT(JJ)-2+K
CTOP(II)=CTOP(JJ)-2+J
ROWCENT(II)=ROWCENT(JJ)-2+J
SLEFT(II)=SLEFT(JJ)-2+K
TLENNERGY=0.0 ;initialize energies
SLENERGY=0.0
SENERGY=0.0
TENERGY=0.0

C The calls to TEST check to see if the input parameters are
C legal, and modifies them if necessary:
C
C 0 < TOP < 257,  (TOP + LENGTH) < 258
C 0 < LEFT < 257,  (LEFT + WIDTH) < 258

CALL TEST(TTOP,TLEFT)
CALL TEST(CTOP(II),SLEFT(II))
Set RDRLK and EUCLID parameters

N=N-1 ; if N=1, compute the energies  
               ; if N=0, compute the distances
IROWCOUNT=0  
TB=INT(FLOAT(TTOP(I)-1)/4.0) ; first template block to be read
SB=INT(FLOAT(TTOP(I)-1)/4.0) ; first scene block
NL=MOD(TTOP(I)-1,4)
N2=MOD((CTOP(I)-1),4)
TLS=LEFT
SLS=LEFT(I)

User check of window parameters

IF(SUPPRESS) GO TO 45
IF(N.EQ.1) GO TO 45

$*begin evaluating the energy (or distances)*$

CORREL=0.0 ; initialize distances
NULDIST=0.0
CALL XRDLK(1, TB, TEMP, 1, IER)
IF (IER .NE. 1) TYPE "1RDRLK "$, TB," error: ", IER
CALL XRDLK(2, SB, SCENE, 1, IER)
IF (IER .NE. 1) TYPE "2RDRLK "$, SB," error: ", IER

This module will continue to loop until the
search windows have been completed (i.e. # of
iterations=(length of the window)/4)

CALL EUCLID(NL, N2, N,$500)

TB=TB+1
SB=SB+1
IF(NL.EQ.0) CALL XRDLK(1, TB, TEMP, 1, IER)
IF (IER .NE. 1) TYPE "1RDRLK "$, TB," error: ", IER
IF(N2.EQ.0) CALL XRDLK(2, SB, SCENE, 1, IER)
IF (IER .NE. 1) TYPE "2RDRLK "$, SB," error: ", IER
CALL EUCLID(NL, N2, N,$500)
IF (NL.EQ.N2) GO TO 110
IF(NL.EQ.0) CALL XRDLK(1, TB, TEMP, 1, IER)
IF (IER .NE. 1) TYPE "1RDRLK "$, TB," error: ", IER
IF(N2.EQ.0) CALL XRDLK(2, SB, SCENE, 1, IER)
IF(IER.NE.1) TYPE"2DDBLK #",SB," error:",IER
GO TO 100

Store the distances computed for iteration II (or
temporarily store the energies if N=1)

500 IF(ENERGY.LT.1.0) ENERGY=1.0
IF(TENERGY.LT.1.0) TENERGY=1.0
IF(S1ENERGY.LT.1.0) S1ENERGY=1.0
IF(T1ENERGY.LT.1.0) T1ENERGY=1.0
S2NORM=SQR(SENERGY)
T2NORM=SQR(TENERGY)
S1NORM=S1ENERGY
T1NORM=T1ENERGY
AREA=FLOAT(LENGTH)*FLOAT(WIDTH)
IF(N.EQ.1) GO TO 40

Compute distance factors

NDISTL1(II)=NLDIST
NL2MIN=T1ENERGY/((SQR(AREA*ENERGY))
NL2MAX=SQR(2.0*(1.0-NL2MIN))
CORPEAK(II)=CORREL/(S2NORM*T2NORM)
CFACTOR(II)=INT(100.0*(CORPEAK(II)-NL2MIN)/(1.0-NL2MIN)+0.5)
IF(CFACTOR(II).LE.0) CFACTOR(II)=0
NL2=SQR(2.0*(1.0-CORPEAK(II)))
L2FACTOR(II)=INT(100.0*(1.0-NL2/NL2MAX)+0.5)
IF(L2FACTOR(II).LE.0) L2FACTOR(II)=0
L1FACTOR(II)=INT(100.0*(1.0-NDISTL1(II)/NL1MAX)+0.5)
IF(L1FACTOR(II).LE.0) L1FACTOR(II)=0

Reset the template window parameters

ITOP=ITOP
ILEFT=ILEFT
WIDTH=WIDTH
LENGTH=LENGTH

600 CONTINUE
TYPE"<>/"********** WRITE RESULTS TO LINEPRINTER **********

C******** User input of comment

DO 625 I=1,200
COMMENT(I)=0

625 CONTINUE
ACCEPT"Enter a 1 to add comment to the output: ",IOPT
IF(IOPT.NE.1) GO TO 650
ACCEPT"Enter # of comment lines (max=4): ",NUMCOM
IF(NUMCOM.LT.1) NUMCOM=1
NUMCOM=MIN(NUMCOM,4)
DO 640 I=1,NUMCOM

143
TY Enter comment line #",",I," to be printed with
$ results between the arrows:

READ(11,9999)COMMENT((50*(I-1)+1))

640 CONTINUE
C Write output header
C
650 WRITE(12,9005)
WRITE(12,9000)
WRITE(12,8000)
IF(REduced)WRITE(12,7500)
WRITE(12,7000)LENGTH,TTOP,TTLENERGY
WRITE(12,6000)INFILE(1),WIDTH,TLT,TENERGY
IF(REduced)WRITE(12,5500)INFILE2(1)
IF(.NOT.,REduced)WRITE(12,5000)INFILE2(1)
WRITE(12,4600)
WRITE(12,4500)
WRITE(12,4200)
WRITE(12,4100)
C Write distance factors
C
II=10
DO 710 JJ=1,NUMPEAKS
DO 700 KK=1,9
II=II+1
A=CFactor(I)
B=L2Factor(I)
C=LLFactor(I)
SCORE=INT((A*B*C)**(1.0/3.0)+0.5)
IF(KK.NE.1)GO TO 698
WRITE(12,4000)JJ,ROWN(JJ),COLUM(JJ),ROWCEN(I),COLCEN(I),
$CTOP(I),SLFT(I),CFactor(I),L2Factor(I),LLFactor(I),
$SCORE
4000 FORMAT(14X,12,"":",14,"",13,5X,13,"",13,6X,13,4X,13,
$7X,13,6X,13,5X,13,5X,13,T132,","
GO TO 700
698 WRITE(12,4050)ROWCEN(I),COLCEN(I),CTOP(I),SLFT(I),
$CFactor(I),L2Factor(I),LLFactor(I),SCORE
4050 FORMAT(30X,13,"",13,6X,13,4X,13,
$7X,13,6X,13,5X,13,5X,13,T132,""
700 CONTINUE
WRITE(12,4051)
4051 FORMAT(" ")
710 CONTINUE
C Write comments to lineprinter
C
IF(IOPT.NE.1)GO TO 800
DO 790 I=1,NUMCOM
WRITE(12,9500)COMMENT((50*(I-1)+1))
790 CONTINUE
WRITE(12,9000)

C Format statements
C
9999 FORMAT(S100)
9500 FORMAT(16X, "COMMENT: ", S100)
9005 FORMAT(11X, "")
9000 FORMAT(11X, "", S100)
8000 FORMAT(12X, "RECOGNITION RESULTS", /*"
7500 FORMAT(15X, "****REDUCED****")
$13,10X,T132, ")
$13,10X,T132, ")
5500 FORMAT(16X, "REDUCED-SCENE FILE --> ", S13, T132, ")
5000 FORMAT(16X, "SCENE FILE --> ", S13, T132, ")
4600 FORMAT(14X, "CORRELATION WINDOW")
4500 FORMAT(14X, "PEAK CENTER TOP LEFT 
\$CORRELATE L2 L1")
4200 FORMAT(14X, "ROW COLUMN 
\$FACTOR FACTOR FACTOR SCORE")
4100 FORMAT(14X, "---" --- "")
3000 TYPE"<7><15><11>**** CHECK LINEPRINTER FOR RESULTS *****<15>"
C
C Present Option Menu ****************************************************
C
GO TO 2010
2002 TYPE"<15>"="Input error. <7> Try again."
2010 TYPE"<15>"="What next?<15>"="Here are the options:"
***="<15>="<11>1 - Try another set of windows"
***="<15>="<11>2 - Start over with new input pictures"
***="<15>="<11>3 - STOP<15>"
ACCEPT"<11>Enter option --> ",&IOPT
IF(IOPT.LE.1.OR.IOPT.GE.3)GO TO 2002
TYPE"<15>"
IF(IOPT.EQ.1)GO TO 3
CALL RESET
IF(IOPT.EQ.2)GO TO 99
STOP
END
C
C Program DISTANCE ****************************************************
SUBROUTINE EUCLID(TSTART, SSTART, N, $)

(Called by DISTANCE) by Lt Jim Cromer

If N=1 —> calculate L1 and L2 energies of template and scene windows
Else —> calculate the normalized L1 and cross-correlation measures between the windows

TSTART, SSTART are the row position within the packed video block of the first row of the window. They are automatically incremented after the first call to EUCLID (TSTART, SSTART between 0-3 inclusive).

REAL NLIDIST, LIDIST, NSUMSQ
INTEGER SCENE(1024), TEMP(1024), SLS, TLS, SSTART, TSTART, WIDTH
COMMON /LIST1/ SCENE, TEMP, SLS, TLS, LIDIST, NLIDIST
COMMON /LIST2/ LENGTH, WIDTH, TRUNCOUNT, SUMSQ, NSUMSQ
COMMON /LIST3/ SLENERGY, TLENERGY, SENERGY, TENERGY
COMMON /LIST4/ S1NORM, S2NORM, T1NORM, T2NORM, CORREL

Set do loop parameters

JMIN=MAX(SSTART, TSTART)
DO 2 J=JMIN, 3
K=TSTART*256+TLS
M=SSTART*256+SLS
RMAX=K+WIDTH-1
IF(N.EQ.1)GO TO 3

Calculate the distances

DO 1 L=K, RMAX
   RSCENE=FLOAT(SCENE(M))
   RTEMP=FLOAT(TEMP(L))
   NLIDIST=ABS((RSCENE/S1NORM)-(RTEMP/T1NORM)) + NLIDIST
   CORREL=(RSCENE*RTEMP)+CORREL
   M=M+1
1 CONTINUE
GO TO 5

Calculate the energies

3 DO 4 L=K, RMAX
   RSCENE=FLOAT(SCENE(M))
   RTEMP=FLOAT(TEMP(L))
   TL1ENERGY=TL1ENERGY+RTEMP
   SL1ENERGY=SL1ENERGY+RSCENE
   SENERGY=(RSCENE**2)+SENERGY
   TENERGY=(RTEMP**2)+TENERGY
   M=M+1
4 CONTINUE
C C Test for the end of the window
  5 IROWCOUNT=IROWCOUNT+1
     IF(IROWCOUNT.GE.LENGTH)RETURN 4
C C Increment block row counters
C
  2 TSTART=TSTART+1
     SSTART=SSTART+1
     IF(SSTART.EQ.4)SSTART=0
     IF(TSTART.EQ.4)TSTART=0
     RETURN
     END
C
C****** Subroutine EUCLID ****************************************************
APPENDIX G:

SUPPORT SUBROUTINES

This appendix contains the following programs:

1. IOF
2. TIMER
3. UNPACK
4. XRDBLK
SUBROUTINE IOF(N,MAIN,F1,F2,F3,MS,S1,S2,S3)
C***********************************************************************
C
Written by Lt. Simmons 10 Sep 1981
Version 2
C
This FORTRAN 5 subroutine will read from the file
COM.COM (FCOM.COM in the foreground) the program name,
any global switches, and up to three local file
names and corresponding local switches.
C
Calling arguments:
C
N is the number of local files and switches to be
read from (F)COM.COM. N must be 1, 2, or 3.
C
MAIN is an ASCII array for the main program file name.
C
F1, F2, and F3 are the three ASCII arrays to return
the local file names.
C
MS is a two-word integer array that holds any global
switches.
C
S1, S2, and S3 are two-word integer arrays that
hold the local switches corresponding to F1 through
F3 respectively.
C
***********************************************************************
C
Dimension the arrays.
C
DIMENSION MAIN(7),MS(2)
INTEGER F1(7),F2(7),F3(7),S1(2),S2(2),S3(2)
C
Check the bounds on N.
C
IF(N.LE.1.OR.N.GT.3)STOP "N out of bounds in IOF."
C
Process the data in (F)COM.COM
C
CALL GROUND(1) ;Find out which ground program is in
IF(I.EQ.0)OPEN 0,"COM.COM" ;Open ch. 0 to COM.COM
IF(I.EQ.1)OPEN 0,"FCOM.COM" ;Open ch. 0 to FCOM.COM
CALL COMARG(0,MAIN,MS,IER) ;Read from (F)COM.COM
IF(IER.NE.1)TYPE" COMARG error:" IER
WRITE(10,1)MAIN(I) ;Type program name
1 FORMAT(' Program ','S13,'running. ')
CALL COMARG(0,F1,S1,IER) ;Read from (F)COM.COM
IF(IER.NE.1)TYPE" COMARG error (F1):" IER
IF(N.EQ.1)GO TO 2 ;Test N
CALL COMARG(0,F2,S2,IER) ;Read from (F)COM.COM
IF(IER.NE.1)TYPE" COMARG error (F2):" IER
IF(N.EQ.2)GO TO 2 ;Test N
CALL COMARG(0,F3,S3,LER) ;Read from (F)COM.COM
IF(LER.NE.1) TYPE" COMARG error (F3):",LER
2 CLOSE 0
RETURN
END

C:
C*********** Subroutine IOF ****************************
SUBROUTINE TIMER(I)

C**********************************************************************
C Subroutine TIMER Written by Lt. Jim Cromer
C Fortran 5
C
This subroutine is used to time the real-time execution time of the calling program. If the parameter passed, I, is equal to 0, the timer is unconditionally started. If I is not equal to 0, the timer is unconditionally stopped, and the total run time is typed on the console CRT.

C Execution Line Format
CALL TIMER(I) ; IF(I.EQ.0), start timing
; IF(I.NE.0), stop timing

C**********************************************************************
COMMON /TIME/ IH1,IM1,IS1
IF(I.NE.0) GO TO 100
CALL FGTIME(IH1,IM1,IS1) ; get starting time
WRITE(10,1000) IH1,IM1,IS1
1000 FORMAT(//" START TIME ---->",I4,"",I3,"",I3)
RETURN
100 CALL FGTIME(IH2,IM2,IS2) ; get stopping time
WRITE(10,2000) IH2,IM2,IS2
2000 FORMAT(//" STOP TIME ---->",I4,"",I3,"",I3)
ITOTAL=3600*(IH2-IH1)+60*(IM2-IM1)+IS2-IS1
HOURS=INT(ITOTAL/3600)
TRON=(ITOTAL-3600*HOURS) ; intermediate variable
MINS=INT(TRON/60)
ISECS=MOD(TRON,60)
WRITE(10,3000) HOURS,MINS,ISECS
3000 FORMAT(//" TOTAL TIME ---->",I4,"",I3,"",I3)
RETURN
END
C********************************************************************** Subroutine TIMER
C Unpacking (packing) routines
C Written by Lt. Simmons           Version 2
C Documented by Lt. Cromer

These subroutines unpack (repack) four 4-bit integers from a
16-bit integer word. The pixels in a video file have to be unpacked if each pixel is to be operated on separately.

Packed video (4 pixels/1 word):

\[
\text{WORD}(N+1) | \text{PIXEL}_1 | \text{PIXEL}_2 | \text{PIXEL}_3 | \text{PIXEL}_4
\]

Unpacked video (4 pixels/4 words):

\[
\begin{align*}
\text{WORD}(X) & | \leftarrow \text{unused} | \text{PIXEL}_1 \\
\text{WORD}(X+1) & | \leftarrow \text{unused} | \text{PIXEL}_2 \\
\text{WORD}(X+2) & | \leftarrow \text{unused} | \text{PIXEL}_3 \\
\text{WORD}(X+3) & | \leftarrow \text{unused} | \text{PIXEL}_4 \\
\end{align*}
\]

where \( N = X \mod 4 \)

SUBROUTINE UNPACK (N, PIXWORD, PIXELS)
INTEGER PIXWORD(N),PIXELS(4,N) ; Four pixels per word
DO 1 I=1,N ; 'N' allows higher-order
   DO 1 J=1,4 ; arrays to be passed,
   PIXELS((5-J),I)=15 AND. PIXWORD(I) ; Pick off right pixel
1 PIXWORD(I)=ISHFT(PIXWORD(I),-4) ; Shift word 4 bits right
RETURN ; to pick off next pixel.
END

SUBROUTINE REPACK(N, PIXELS, PXWD)
INTEGER PIXELS(4,N),PXWD(N)
DO 1 J=1,N
   PXWD(J)=0
   DO 1 I=1,4
   PXWD(J)=ISHFT(PXWD(J),4)
1 PXWD(J)=PIXELS(I,J)+PXWD(J)
RETURN
END

C************ Packing Subroutines ****************************
SUBROUTINE XRDBLK(CH,J,FILE,I,IER)

C******************************************************************************
C by Lt. Jim Cromer
C Subroutine XRDBLK performs a RDBLK to the designated
C channel, reads a packed video file block, and
C returns an unpacked array.
C******************************************************************************

INTEGER CH,FILE(1024),VIDEO(256)
K=256*I
IF(J.GE.0.AND.J.LE.63)GO TO 1
TYPE*ERROR: <7> BLOCK POINTER OUT OF BOUNDS IN XRDBLK*
TYPE*     J=",J
STOP
1 IF(I.EQ.1)GO TO 2
TYPE*ERROR IN <7>XRDBLK"
TYPE* # Blocks to be read="I
STOP
2 CALL RDBLK(CH,J,VIDEO,I,IER)
DO 3 L=1,K
DO 3 M=1,4
   ICOUNT=5-M+(L-1)*4
   FILE(ICOUNT)=15.AND. VIDEO(L)
   VIDEO(L)=ISHFT(VIDEO(L),-4)
3 CONTINUE
RETURN
END

C******** Subroutine XRDBLK ********************************************
APPENDIX H: PRELIMINARY RESULTS

Summary of Tank SCOREs

Template and Tank windows both globally normalized

<table>
<thead>
<tr>
<th>Template</th>
<th>Scene</th>
<th>High</th>
<th>Low</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTEMPH3</td>
<td>PTANKH3</td>
<td>35</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>RTEMPH3</td>
<td>RPTANKH3</td>
<td>36</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>PTEMPD4</td>
<td>PTANKB2</td>
<td>41</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>PTEMPD4</td>
<td>PTANKC3</td>
<td>49</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>PTEMPD4</td>
<td>PTANKE2</td>
<td>56</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>PTEMPD4</td>
<td>PTANKG4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>36</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

Template and Tank windows both grid normalized (9x5 grid used)

<table>
<thead>
<tr>
<th>Template</th>
<th>Tank</th>
<th>High</th>
<th>Low</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMD4</td>
<td>NORMB2</td>
<td>65</td>
<td>40</td>
<td>52</td>
</tr>
<tr>
<td>' '</td>
<td>NORMC3</td>
<td>69</td>
<td>43</td>
<td>58</td>
</tr>
<tr>
<td>' '</td>
<td>NORMD2</td>
<td>64</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>' '</td>
<td>NORME2</td>
<td>67</td>
<td>45</td>
<td>57</td>
</tr>
<tr>
<td>' '</td>
<td>NORMG4</td>
<td>42</td>
<td>16</td>
<td>29</td>
</tr>
<tr>
<td>' '</td>
<td>NORMH3</td>
<td>79</td>
<td>24</td>
<td>43</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>64</td>
<td>35</td>
<td>49</td>
</tr>
</tbody>
</table>
Summary of Scene SCOREs

Template and Scene windows both globally normalized

<table>
<thead>
<tr>
<th>Template</th>
<th>Scene</th>
<th>High</th>
<th>Low</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTEMPH3</td>
<td>PSCENEI3</td>
<td>16</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>PTEMPH3</td>
<td>PSCENE04</td>
<td>24</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>RPTEMPH3</td>
<td>RPSCENE04</td>
<td>20</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>RPTEMPH3</td>
<td>RPSCENEL1</td>
<td>19</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>19</td>
<td>14</td>
<td>17</td>
</tr>
</tbody>
</table>

Template and Scene windows both grid normalized (9x5)

<table>
<thead>
<tr>
<th>Template</th>
<th>Scene</th>
<th>High</th>
<th>Low</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMD4</td>
<td>WHITE</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NORMD4</td>
<td>NORM7</td>
<td>15</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
James H. Cromer was born on 10 April 1959 in Cleveland, Ohio. He graduated from Kenston High School, Chagrin Falls, Ohio, in 1977. He attended Grove City College, Grove City, Pennsylvania, on a 4-year AFROTC scholarship. He received the Bachelor of Science degree in Electrical Engineering, and was commissioned a Second Lieutenant in the United States Air Force, in May 1981. He entered the School of Engineering, Air Force Institute of Technology, in June 1981, and was chosen to receive the 1981-1982 IEEE Outstanding Student Award in May 1982. He is a member of Tau Beta Pi, Sigma Pi Sigma, and Eta Kappa Nu.

Permanent Address: 204 E. Mill St.
Port Allegany, PA 16743
Scene Analysis: Non-linear Spatial Filtering for Automatic Target Detection

James H. Cromer
2nd Lt USAF

Air Force Institute of Technology (AFIT/EN)
Wright-Patterson AFB, Ohio 45433

December 1982

Approved for public release; distribution unlimited

Cross-correlation Pattern Recognition
Form Analysis Scene Analysis
Image Processing Target Classification
Energy Normalization Target Detection

This work focuses on a method for two-dimensional pattern recognition. The method includes a global search scheme for candidate windows of interest, based on Fourier domain cross-correlation. A method to normalize the input scene by local rectangular regions, in an attempt to efficiently approximate search window normalization, is presented. Also developed is a candidate window (potential target) similarity measure, based on the normalized Li and Euclidean distances, which is independent of the template DC value and its energy. Observations on the performance of the algorithm applied to visual...
spectrum photographs of tanks in a realistic environment are included. Also included is the software needed to implement the algorithm on Data General Eclipse S/250 minicomputer.