DIGITAL IMAGE REGISTRATION METHOD
BASED UPON BINARY BOUNDARY MAPS

by R. R. Jayroe, J. F. Andrus, and C. W. Campbell

George C. Marshall Space Flight Center
Marshall Space Flight Center, Ala. 35812
### LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Boundary map from a portion of ERTS imagery</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Computation of $S_x$ and $S_y$ for channel $K$</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>Subcase for DO 56 loop</td>
<td>7</td>
</tr>
<tr>
<td>4.</td>
<td>Subcase for DO 68 loop, $JL &gt; 0$</td>
<td>7</td>
</tr>
<tr>
<td>5.</td>
<td>Subcase for DO 68 loop, $JL \leq 0$, $NSPAN \leq NU$, $JP(L) &gt; JW(K)$</td>
<td>8</td>
</tr>
<tr>
<td>6.</td>
<td>Subcase for DO 78 loop, $JL \leq 0$, $JP(L) \leq JW(K)$</td>
<td>9</td>
</tr>
<tr>
<td>7.</td>
<td>Subcase for DO 78 loop, $JL \leq 0$, $JP(L) &gt; JW(K)$, $NSPAN &gt; NU$</td>
<td>10</td>
</tr>
<tr>
<td>8.</td>
<td>Window selection</td>
<td>12</td>
</tr>
<tr>
<td>9.</td>
<td>Overlay of digital images</td>
<td>12</td>
</tr>
<tr>
<td>10.</td>
<td>Binary boundary map data set</td>
<td>16</td>
</tr>
<tr>
<td>11.</td>
<td>Binary correlation output for a 32 by 32 window array</td>
<td>17</td>
</tr>
<tr>
<td>12.</td>
<td>Correlation lag array size versus running time</td>
<td>18</td>
</tr>
<tr>
<td>13.</td>
<td>Number of variables needed to define window array</td>
<td>20</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Number of Occurrences of Each Column Shift for Figure 4</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>Number of Occurrences of Each Column Shift for Figure 5</td>
<td>9</td>
</tr>
<tr>
<td>3.</td>
<td>Number of Occurrences of Each Column Shift for Figure 6</td>
<td>10</td>
</tr>
<tr>
<td>4.</td>
<td>Number of Occurrences of Each Column Shift for Figure 7</td>
<td>11</td>
</tr>
<tr>
<td>5.</td>
<td>Running Time Versus Correlation Lag Array Size</td>
<td>18</td>
</tr>
<tr>
<td>6.</td>
<td>Compression Factor for Window Array</td>
<td>21</td>
</tr>
</tbody>
</table>
FOREWORD

This study was initiated under the direction of Dr. R.R. Jayroe, Flight Data Statistics Office, Aerospace Environment Division, Aero-Astrodynamics Laboratory, Marshall Space Flight Center. This work was accomplished via the in-house efforts of Dr. R.R. Jayroe and Mr. C.W. Campbell of the same office, and by the efforts of Dr. J.F. Andrus of Northrop Services, Inc., Huntsville, Alabama, under contract NAS8-21810. Dr. Andrus must be credited for conceiving the original idea used in the binary correlation routine, and has continued to contribute to this study since joining the faculty of Louisiana State University, New Orleans, in August 1973.
DIGITAL IMAGE REGISTRATION METHODS BASED UPON
BINARY BOUNDARY MAPS

I. INTRODUCTION

The ability to register or match the ground scene of images in the form of digital data serves an important role in the analysis of remotely sensed, multispectral, earth observation data and change detection. In the case of multispectral camera data, the images from the different camera stations must be registered before a proper analysis can be performed. Registration is also necessary when the data are acquired from several different sensors. In the case of change detection, where data are acquired from the same ground scene but at different time intervals, registration has an important role in determining what changes in shape have taken place in the ground scene, and registration permits an analysis to determine how the multispectral signatures of various ground features change as a function of time. For meteorological applications, registration methods based on meteorological satellite imagery can be used to estimate cloud velocities for global weather information.

Because of the typically large volumes of data involved, the use of binary maps in registration is recommended for the reasons discussed below. Converting the raw data to a binary boundary map typically represents a data compression of 60 to 70 percent of the original data. An additional significant compression of data is realized by working with sequences of, rather than individual, boundary points, which requires only the knowledge of the start scan and column and the length of the sequence.

If $A_1(x,y)$ and $A_2(x+\xi, y+\eta)$ represent the amplitudes of the data in images 1 and 2 respectively, at scan coordinates $x$ and $x+\xi$ respectively, and at column coordinates $y$ and $y+\eta$ respectively, then the average product $\overline{A_1(x,y) A_2(x+\xi, y+\eta)}$ is computed over $x$ and $y$ for various combinations of $\xi$ and $\eta$ to determine the scan and column shifts necessary to register the two digital images. If this average is applied to the raw data, it may also be necessary to remove mean values and normalize in order to produce a correlation coefficient with a well defined peak. The use of the raw data appears to have some drawbacks as compared to using binary border maps. First, an ambiguity can result, since a large negative correlation peak can occur and represent a good match as well as a large positive peak. This is in most part due to the variation of the data acquired under different environmental conditions; i.e., different sun angles, seasons, atmospheric conditions, etc. On the other hand, binary boundary maps are produced from relative changes in the data, and the boundaries indicated in the ground scene tend not to change with environmental conditions. Since the binary boundary maps contain data that consist only of 0's and 1's, the average product $\overline{A_1(x,y) A_2(x+\xi, y+\eta)}$ will always be positive and tends to be sharp so that removal of means and normalization is not necessary. In addition, it is possible to compute the average product $\overline{A_1(x,y) A_2(x+\xi, y+\eta)}$ without multiplying when binary data are used. Addition can be used to replace multiplication, and this reduces computer computation time significantly.
Section II is a discussion of the production of binary boundary maps, while Sections III and IV are concerned with the correlation and registration schemes, respectively. Section V contains the results, test cases, and summary.

II. BINARY BOUNDARY MAPS

The purpose of the binary boundary map is to categorize the digital data representing the ground scene into homogeneous areas and boundaries, and an example of a boundary map from ERTS digital imagery with some feature identification is shown in Figure 1.

The computer program used to generate the boundary map typically runs about 13 minutes (IBM-7094 time) on an area of 1000 scans by 255 columns, which is approximately 327 samples/second. This time includes the amount of time required to read and compile the program in addition to the calculation time. At present, no concerted effort has been made to optimize the running time of this program or to specialize its output for the registration program, but the efforts are in a holding status for future consideration.

The logic of the program is to work with two scans of raw digital data simultaneously. Let \( kx_{ij} \) represent the algebraic value of the data in the \( k \)th channel of data at scan \( i \) and column \( j \) as shown in Figure 2. If there are \( n \) channels of data, then \( k \) ranges from 1 to \( n \). For each location \( i, j \), the average vector component distances, which represent various degrees of spectroscopic changes within the ground scene, are computed using the formulas

\[
S_x = \left[ \frac{1}{n} \sum_{k=1}^{n} (kx_{i,j} - kx_{i-1,j})^2 \right]^{1/2}
\]

Figure 1. Boundary map from a portion of ERTS imagery.
and

\[
S_y = \left[ \frac{1}{n} \sum_{k=1}^{n} (k^x_{i,j} - k^y_{i,j-1})^2 \right]^{\frac{1}{2}}
\]  \hspace{1cm} (1)

and are stored in a joint histogram, \( P(S_x, S_y) \). Dividing by \( n \) in equation (1) assures that \( S_x \) and \( S_y \) have the same typical range regardless of the number of data channels. Based upon previous experience with various types of data, it was found that the size of the joint histogram array could be limited in size to a 50 by 50 array such that any data element with a value of \( S_x \) or \( S_y \) larger than 50 could be automatically assigned as a boundary element. As the raw data are read into the program and the number of occurrences of \( S_x \) and \( S_y \) are accumulated in the joint histogram, a new tape is created for later use. This tape contains the numbers 1 through \( 50^2 = 2500 \) and instead of writing \( S_x \) and \( S_y \) on this tape for each data element, one number is written on the tape that gives the location of \( S_x \) and \( S_y \) in the joint histogram. The location, \( l \), of \( S_x \) and \( S_y \) in the joint histogram is computed using

\[
l = (51) S_x + S_y
\]  \hspace{1cm} (2)

and is a unique one-dimensional representation of the two dimensions \( S_x \) and \( S_y \). The new tape containing a number \( l \) for each data element eliminates the necessity for recalculating \( S_x \) and \( S_y \) for each data element a second time. After all the data elements from the data set have been exhausted and the joint histogram is complete, a decision is made as to which combinations of \( S_x \) and \( S_y \) are to be considered as boundaries. The decision curve for a boundary element is based upon the formula

![Figure 2. Computation of \( S_x \) and \( S_y \) for channel K.](image-url)
\[
\left( \frac{S_x}{S_x' + ASCN} \right)^{IPOW} + \left( \frac{S_y}{S_y' + ACOL} \right)^{IPOW} > (2)BLIM \quad (3)
\]

where \(S_x'\) and \(S_y'\) are the values of \(S_x\) and \(S_y\) at the mode of the joint histogram and ASCN, ACOL, IPOW, and BLIM are input parameters to the program. The input parameters allow for a wide variety of decision curve shapes and positions. Nominally IPOW = 2 and BLIM = 1, whereas ASCN is usually equal to ACOL and must be estimated based upon experience. The possible values of 1 or correspondingly \(S_x\) and \(S_y\) are then inserted into equation (2) and the variable N(I) in the computer program is set equal to 0 if equation (2) is not satisfied and is set equal to 1 if equation (2) is satisfied. The tape containing I for each data element is then read into the program, and the value of N(I) is written out as another tape which is called the binary boundary map and contains only the numbers 0 and 1. The use of the intermediate tape containing the I values permits the boundary decision curve to be varied without recalculating \(S_x\) and \(S_y\) for every data element.

III. CORRELATION SCHEME

The development work for the correlation scheme was initiated because of a lack of a readily available registration program and after a review and discussion of some of the most recently published papers on registration, such as References 1 through 3. A computer listing of the correlation scheme is provided in Appendix A. The central idea is to compute only those correlation delays which are affected by overlapping boundary elements for each shift of the window against the picture.

Before describing the correlation scheme, it is necessary to define the computer program variables and terminology for the computer routine.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td>A section of a boundary map to be used as the reference data.</td>
</tr>
<tr>
<td>Window</td>
<td>A section of a boundary map which is to be registered with the picture. The window has one-half as many scans and columns of data as the picture.</td>
</tr>
<tr>
<td>NROW</td>
<td>The number of data scans in the window.</td>
</tr>
<tr>
<td>NCOL</td>
<td>The number of data samples per scan or columns of data in the window. Nominally, NROW is equal to NCOL.</td>
</tr>
</tbody>
</table>
The average product $A_1(x,y) A_2(x+1, y+1)$. The value of NCOR(I,J) is the number of coinciding boundary points that occur when the upper left corner element of the window is placed upon the $i$th row and $j$th column of the picture. ($i = 1, 2, ..., NROW + 1$ and $j = 1, 2, ..., NCOL + 1$.)

KMAX

Total number of boundary sequences in the window.

LMAX1

Total number of boundary sequences in the picture for $i = 1, 2, ..., NROW$.

LMAX

Total number of boundary sequences in the picture for all $i$, $i = 1, 2, ..., 2$ NROW.

NW(J)

Dummy variable for reading binary boundary map data. ($J = 1, ..., NROW$ for window; $J = 1, ..., 2$ NROW for picture.)

K

Number of the $k$th window boundary sequence ($k = 1, ..., KMAX$).

L

Number of the $l$th picture boundary sequence ($l = 1, ..., LMAX$).

IW(K)

The value of IW(K) is the scan number minus one of the $k$th boundary sequence in the window.

JW(K)

The value of JW(K) is the column number minus one of the beginning of the $k$th boundary sequence in the window.

NSQW(K)

The value of NSQW(K) is the number of boundary elements minus one in the $k$th boundary column sequence in the window.

IP(L)

The value of IP(L) is the scan number of the $l$th boundary sequence in the picture.

JP(L)

The value of JP(L) is the column number of the beginning of the $l$th boundary sequence in the picture.

NSQP(L)

The value of NSQP(L) is the number of boundary elements minus one in the $l$th boundary column sequence in the picture.

I

The value of I is the scan shift in NCOR(I,J).
The value of JL is the lower value of the column shift J in NCOR(I,J).

The value of JU is the upper value of the column shift J in NCOR(I,J). (J = JL, JL + 1, ..., JU.)

The value of NAD is the number of coinciding boundary points at a given I and J for the Kth window boundary sequence and the Lth picture boundary sequence.

The value of JM controls the value of NAD. JM causes NAD to decrease as the Kth and Lth boundary sequences decreasingly overlap because of an increase in J from JL to JU.

For a given I, the value of NU is the column number of the end of the Lth boundary sequence in the picture.

For a given I, the value of NSSPAN is the column number of the end of the Kth boundary sequence in the window.

MINL is a dummy variable which changes and represents the smaller of the window or picture boundary sequence minus one.

CALL TIMNOW( ) Subroutine for calling internal clock to time computation of NCOR(I,J).

The first step of the program is to read in the sections of the binary boundary maps to be considered, skipping all of the nonboundary or zero elements, and storing the start scan, start column, and column length of the sequences in the window and picture. This information is stored consistent with the definition of IW(K), JW(K), NSQW(K), IP(L), JP(L), NSQP(L), KMAX, LMAX1, and LMAX, and results in a considerable compression of the data. The calculation of the above variables is accomplished in the DO 20 nested do-loop for the window and the DO 40 nested do-loop for the picture. The remaining part of the program contains two major calculation segments, the DO 100 nested do-loop and the DO 200 nested do-loop. Because of their similarity it will suffice to explain only one of these do-loops and then point out their differences. The DO 100 segment is concerned with the entire window and the top half of the picture; i.e., I = 1, 2, ..., NROW. The two outer do-loops, DO 100 and DO 80, determine the I and J shifts that cause the individual boundaries of the K window sequences to coincide with the individual boundaries of the L picture sequences. No negative shifts are permitted; i.e., 1 ≤ I ≤ NCOL + 1 and 1 ≤ J ≤ NCOL + 1. Within the DO 100 and DO 80 loops, there are 3 cases to be considered. The first case is the DO 50 loop and it occurs when the minimum length, MINL, of either the window or picture sequence is one boundary element. This occurrence is illustrated in Figure 3 and can result in an addition of at most 1 to NCOR(I,J) for each value of J. The case which probably occurs most often is
the DO 68 loop, and a typical example is shown in Figure 4. The first time through the DO 68 loop NAD = 1 is added to NCOR(I,J) and NAD continues to increase by one until it is greater than MINL, the minimum length of either sequence minus one. Once NAD is one greater than MINL it remains that value until J is greater than JM, and then NAD decreases by one each time until the two sequences no longer overlap in the column direction. Thus, both the initial and final values of NAD added to NCOR(I,J) are one, while the maximum value of NAD is the number of boundary elements in the smaller of the two sequences. Table 1 is a convenient way of viewing the number of occurrences of each column shift due to the presence of both sequences. The top two rows list the possible values of J or the column shift, while the next three rows list the possible values
TABLE 1. NUMBER OF OCCURRENCES OF EACH COLUMN SHIFT FOR FIGURE 4

<table>
<thead>
<tr>
<th></th>
<th>JL</th>
<th>JM</th>
<th>JU</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP(L)</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + 1</td>
<td>JL+1</td>
<td>JL+2</td>
<td>JL+3</td>
</tr>
<tr>
<td>JP(L) + NSQP(L)</td>
<td>JL+2</td>
<td>JL+3</td>
<td>JL+4</td>
</tr>
<tr>
<td>NAD</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

of J for each boundary element in the picture. The last row is the corresponding value for NAD for each J or the total number of occurrences of each column shift. An additional subcase for the DO 68 loop can occur when JL is negative, JP(L) > JW(K), and the length of the window sequence is equal to or less than the picture sequence. An example of this subcase is shown in Figure 5 which corresponds to Table 2.

The final two subcases occur in connection with the DO 78 loop. Both subcases have the minimum column shift, JL, which is initially negative. The first subcase has JP(L) less than or equal to JW(K) and is illustrated in Figure 6 which corresponds to Table 3. The second subcase is illustrated in Figure 7 which corresponds to Table 4.

Figure 5. Subcase for DO 68 loop, JL < 0, NSPAN < NU, JP(L) > JW(K).
TABLE 2. NUMBER OF OCCURRENCES OF EACH COLUMN SHIFT FOR FIGURE 5

<table>
<thead>
<tr>
<th></th>
<th>JL</th>
<th>JM</th>
<th>JU</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP(L)</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + 1</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + 2</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + NSQP(L)</td>
<td>JL+1</td>
<td>JL+2</td>
<td>JL+3</td>
</tr>
<tr>
<td>NAD</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 6. Subcase for DO 78 loop, JL ≤ 0, JP(L) ≤ JW(K).
TABLE 3. NUMBER OF OCCURRENCES OF EACH COLUMN SHIFT FOR FIGURE 6

<table>
<thead>
<tr>
<th>JP(L) + 1</th>
<th>JL</th>
<th>JM</th>
<th>JU</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP(L) + 2</td>
<td>JL</td>
<td>JL+1</td>
<td></td>
</tr>
<tr>
<td>JP(L) + 3</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + NSQP(L)</td>
<td>JL+1</td>
<td>JL+2</td>
<td>JL+3</td>
</tr>
<tr>
<td>NAD</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 7. Subcase for DO 78 loop, JL < 0, JP(L) > JW(K), NSPAN ≥ NU.
TABLE 4. NUMBER OF OCCURRENCES OF EACH COLUMN SHIFT FOR FIGURE 7

<table>
<thead>
<tr>
<th></th>
<th>JL</th>
<th>JM</th>
<th>JU</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP(L)</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + 1</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>JP(L) + NSQP(L)</td>
<td>JL</td>
<td>JL+1</td>
<td>JL+2</td>
</tr>
<tr>
<td>NAD</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The second segment of the program or the DO 200 loop is identical to the first segment of the program, the DO 100 loop, except for the statement numbers. The main difference is that the DO 200 loop, while considering the entire window, only considers the bottom half of the picture. Thus, the DO 100 loop considers less and less of the first half of the picture sequences for all of the window sequences, and the DO 200 loop considers less and less of the window for the last half of all of the picture sequences. In the DO 100 loop, the scan shift \( I \) starts at the maximum available value and decreases to \( I = 1 \), while in the DO 200 loop \( I \) starts at the minimum available value and increases to a maximum of \( I = NROW + 1 \). A scan shift of \( I = 1 \) and a column shift of \( J = 1 \) indicate that the window and picture are already registered.

IV. REGISTRATION SCHEME

In applying the registration scheme it is assumed that the data can be misaligned by translation and/or rotation, but that no distortion exists between the two digital images. If significant distortions do exist over the entire images, then it may become necessary to register subportions of the digital images as separate data sets.

It is also assumed that both digital images are of the same scale. If the images are not of the same scale, then it is possible to scale down one image by locating several corresponding boundary elements on both images, choose a corresponding reference boundary element on each image, and compute for each image the average distance of all the other boundary elements from the reference boundary element. The ratio of the two average distances gives a scaling factor. The raw data image with the larger average distance can then be scaled to the same size as the other image, and a new boundary map can be computed.
To register two boundary maps, choose one map as the reference map or picture and choose subportions of the other map to be used as windows as illustrated in Figure 8. Figure 9 shows a portion of the two maps overlayed and the minimum and maximum scan and column shifts for the window. For the registration to work properly, the final

Figure 8. Window selection.

Figure 9. Overlay of digital images.
two assumptions are that the portions of the picture corresponding to the window are within the allowable range of the window and that the amount of rotation present in the picture or the window can be neglected as far as the computation of NCOR(1, J) is concerned and can be treated as a local translation. If the latter assumption is not valid, it may be necessary to repeat the registration procedure again after the data have been translated and/or rotated based upon the first registration effort. In the event that the first assumption is correct, it should be possible to overcome the second assumption by repeated application of the registration procedure to the data.

The mathematics of the translation-rotation required to register the two images is given below. Let \( w_x^i, w_y^i \) be the scan and column coordinates respectively of the upper left corner on the \( i \)th window in the image to be registered, and let \( l_1 \) and \( J_1 \) correspond to the maximum value of NCOR(1, J) for the \( i \)th window. For any one of the windows, for example the \( j \)th, let the coordinates

\[
p_x^j = w_x^j + l_j - 1
\]

and

\[
p_y^j = w_y^j + J_j - 1
\]

on the picture be designated as the center of rotation. Thus, if the entire window image is shifted by the amounts \( l_1 \) and \( J_1 \), the coordinates \( (p_x^j, p_y^j) \) and \( (w_x^j, w_y^j) \) will coincide and the rest of the image can be rotated about these coordinates to complete the registration, provided a rotation exists. If no rotation exists, the images are registered and the maximum value of NCOR(1, J) for the rest of the window occurs at \( l = J = 1 \). Assuming that rotation does exist, it will be necessary to compute the following quantities.

\[
\bar{p_x} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - p_x^j) (w_x^j - w_x^i)
\]

and

\[
\bar{p_y} = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - p_y^j) (w_y^j - w_y^i)
\]
and

\[ \bar{p}^x w^x = \frac{1}{N-1} \sum_{i \neq j} (p^x_i - p^x_j)(w^y_i - w^y_j) \quad (7) \]

\[ \bar{w}^x p^y = \frac{1}{N-1} \sum_{i \neq j} (w^x_i - w^x_j)(p^y_i - p^y_j) \quad (8) \]

and

\[ \tan \theta = \frac{\bar{p}^x w^y - \bar{w}^x p^y}{\bar{p}^x w^x + \bar{w}^y p^y} \quad (9) \]

where \( N \) is the total number of windows. To fetch the data from the image to be registered, it is first necessary to solve for \( \theta \), and for each picture coordinate \((p^x, p^y)\), it is necessary to solve the equations,

\[ p^x \cos \theta + p^y \sin \theta - p^x_j \cos \theta - p^y_j \sin \theta + p^x_j = w^x \]

and

\[ -p^x \sin \theta + p^y \cos \theta - p^x_j \sin \theta - p^y_j \cos \theta + p^y_j = w^y \]

for the appropriate unregistered image coordinates. In general, the coordinates \((w^x, w^y)\) will not be integers, but in most cases it is probably most appropriate and less time consuming to round off to the nearest integer coordinate rather than try to interpolate the value of the data between the original data points.

The registration routine is currently being programmed and debugged by MSFC’s Computation Laboratory, Engineering Computation- Division (S&E-COMP-RRV). The routine has not yet been modified to accept the correlation scheme, but instead accepts a manual input of corresponding boundary elements from two different images.
V. TEST CASES, RESULTS, AND SUMMARY

The binary correlation routine was compared to a correlation and Fast Fourier Transform (FFT) correlation routine by obtaining running times from an internal clock on the IBM-7044 computer. A listing of the correlation routine and a program for using the FFT correlation routine are presented in Appendices B and C. The Fast Fourier Transform used, FFT7, is not presented in this report since it has been evaluated and listed in Reference 4 along with nine other FFT routines. According to Reference 4, FFT7 was the most versatile and ranked third in speed to the fastest routine, FFT3, which was twice as fast but worked only on real data, performed only a (+i) transform, and produced only a half transform. The second fastest transform was FFT8, which was only 25 milliseconds faster than FFT7.

The data set on which the programs were run is shown in Figure 10. In the data set, only the locations of the boundaries are printed out and represented as 1’s, and the window and the picture both start in the upper left corner. The size of the data array used for the window is the first NCOL columns by NROW scans, and the picture is twice as large. For the Fast Fourier Transform the window has to be the same size as the picture, and consequently zeros had to be added to the window array. The window and picture data arrays were arranged such that they were already in registration or such that the maximum value of NCOR(I,J) occurred at I = J = 1.

The output of the binary correlation routine for a 32 by 32 window array is shown in Figure 11, and the maximum number of NCOR(I,J) is the total number of boundary elements in the window. Figure 12 is a graph of correlation lag array size versus running time for the binary correlation, Fast Fourier Transform correlation, and correlation routines. The actual running times for each program versus correlation array size are listed in Table 5. Storage requirements for the FFT correlation routine were such that it was not possible to compute 41 by 41 correlation lag points or larger, and the irregular running times as shown in Figure 12 are a result of the FFT routine being more efficient for some array sizes than others. The correlation routine was able to handle 57 by 57 correlation lag points, but the running time was already 5 minutes for 41 by 41 correlation lag points. Since it appears that running times for the correlation routine can easily be extrapolated from Figure 12 and Table 5, no larger size arrays were run.

According to Reference 1, the Sequential Similarity Detection Algorithm (SSDA) mentioned therein is approximately 50 times faster than the FFT correlation method. The factor of 50 is estimated from time ratios relating arithmetic operations on the IBM 360/65 used in computing the numerator of the correlation coefficient. By relating all arithmetic operations to integer adds for a 32 by 32 window and a 256 by 256 picture, the equivalent integer adds for the correlation, FFT correlation, and SSDA respectively were $2.57 \times 10^8$ and $2.2 \times 10^6$, which indicate a factor of 100 times faster than the FFT correlation. To compare the binary correlation with the SSDA it was necessary to put a counter in the DO loops containing NCOR(I,J) and to use the formula of
Figure 10. Binary boundary map data set.
**Figure 11.** Binary correlation output for a 32 by 32 window array.
Figure 12. Correlation lag array size versus running time.

**TABLE 5. RUNNING TIME VERSUS CORRELATION LAG ARRAY SIZE**

<table>
<thead>
<tr>
<th>Lag Array Size</th>
<th>Running Time in $60^1$ seconds/Times Slower Than Binary Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 by 25</td>
<td>58/10.09</td>
</tr>
<tr>
<td>29 by 29</td>
<td>124/10.22</td>
</tr>
<tr>
<td>33 by 33</td>
<td>230/4.27</td>
</tr>
<tr>
<td>37 by 37</td>
<td>381/4.67</td>
</tr>
<tr>
<td>41 by 41</td>
<td>559</td>
</tr>
<tr>
<td>45 by 45</td>
<td>850</td>
</tr>
<tr>
<td>49 by 49</td>
<td>1167</td>
</tr>
<tr>
<td>53 by 53</td>
<td>1710</td>
</tr>
<tr>
<td>57 by 57</td>
<td>2377</td>
</tr>
<tr>
<td>81 by 81</td>
<td>8371</td>
</tr>
<tr>
<td>101 by 101</td>
<td>19761</td>
</tr>
</tbody>
</table>
Reference 1 for computing the equivalent number of integer adds for a 32 by 32 window and 64 by 64 picture. The formula presented in Reference 1 is $4(1 + 10\sqrt{M/32})(L - M+1)^2$ equivalent integer adds, where L and M are the picture and window length or width, respectively. For the above mentioned picture and window the number of equivalent integer adds for the SSDA is 47,916 while the number of integer adds for the particular data set shown in Figure 10 is 20,804. Thus, it would appear that the binary correlation could possibly be 2.3 times faster than the SSDA. However, it is also possible that computing equivalent integer adds for various types of programs does not give the complete story, since it is necessary in all the computer programs to execute additional computer statements other than updating the correlation. Otherwise, the binary correlation would be approximately 200 times faster than the FFT correlation. Table 5 indicates that it is not. It is realized that one does not obtain "something for nothing," and the processing time needed to produce a binary boundary map could negate the speed of the binary correlation. However, two points in addition to those mentioned in Section I are worth considering. According to Reference 1 it may become necessary to resort to the use of boundary maps when the channels under consideration are widely separated in spectral wavelength. This is certainly evidenced when one is working with near infrared or thermal imagery which tends to be quite noisy. Secondly, Figure 13 suggests a way to optimize the production of a boundary map.

Figure 13 illustrates the number of variables needed to define various window sizes for the correlation and binary correlation methods using the data set illustrated in Figure 10. Also included in Figure 13 is the number of variables needed to define the location of the boundary elements only in the data set, which is twice the number of boundary points. For the correlation it is necessary to store the entire window array, while for the binary correlation it is necessary only to store $3 \times \text{KMAX}$ variables, the start scan, the start column, and the sequence length for each boundary sequence. Thus, Figure 13 indicates the amount of data compression that is possible. Table 6 lists the compression factor for various window sizes, which is the total number of data points divided by $3 \times \text{KMAX}$.

The production of a boundary map could be optimized by recognizing that three variables are needed to define each boundary sequence. Also, from the logic used in calculating $\text{IW(K)}$, $\text{JW(K)}$, $\text{NSQW(K)}$, $\text{IP(L)}$, $\text{JP(L)}$, and $\text{NSQP(L)}$, it is only necessary to determine boundaries within a scan and not necessary to determine boundaries that occur across two adjacent scan lines. Thus, the boundary program described in Section II could be optimized by working with one scan of data at a time instead of two and by eliminating all boundary sequences that contain less than three boundary elements. This would also optimize the compression by assuring that $3 \times \text{KMAX}$ is always equal to or less than the number of boundary elements.

It is foreseen that in some cases it may become necessary to subtract mean values from $\text{NCOR(I,J)}$ and normalize to produce a proper registration of two data sets. In this respect it is interesting to examine the correlation coefficient in terms of rewards and
TOTAL NUMBER OF VARIABLES NEEDED TO DEFINE DATA SET

FOR CORRELATION

2 X NUMBER OF BOUNDARY POINTS

3 X KMAX

TOTAL NUMBER OF DATA POINTS

Figure 13. Number of variables needed to define window array.

penalties used in the binary correlation method. The binary correlation method presented in Section III only rewards for boundary elements that match and does not penalize for mismatches of boundary and nonboundary elements. By subtracting mean values and normalizing, it is possible to obtain rewards, for matching boundary elements and matching nonboundary elements, and penalties for mismatches of boundary and nonboundary elements. Before examining the correlation coefficient, it will be necessary to define a few terms.
TABLE 6. COMPRESSION FACTOR FOR WINDOW ARRAY

<table>
<thead>
<tr>
<th>Window Array Size</th>
<th>Number of Boundary Elements</th>
<th>KMAX</th>
<th>Compression Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 by 24</td>
<td>61</td>
<td>33</td>
<td>5.81</td>
</tr>
<tr>
<td>28 by 28</td>
<td>90</td>
<td>48</td>
<td>5.44</td>
</tr>
<tr>
<td>32 by 32</td>
<td>122</td>
<td>70</td>
<td>4.88</td>
</tr>
<tr>
<td>36 by 36</td>
<td>158</td>
<td>95</td>
<td>4.55</td>
</tr>
<tr>
<td>40 by 40</td>
<td>183</td>
<td>115</td>
<td>4.64</td>
</tr>
<tr>
<td>44 by 44</td>
<td>237</td>
<td>151</td>
<td>4.27</td>
</tr>
<tr>
<td>48 by 48</td>
<td>296</td>
<td>178</td>
<td>4.31</td>
</tr>
<tr>
<td>52 by 52</td>
<td>368</td>
<td>229</td>
<td>3.94</td>
</tr>
<tr>
<td>56 by 56</td>
<td>450</td>
<td>281</td>
<td>3.72</td>
</tr>
<tr>
<td>64 by 64</td>
<td></td>
<td>355</td>
<td>3.85</td>
</tr>
<tr>
<td>72 by 72</td>
<td></td>
<td>451</td>
<td>3.83</td>
</tr>
<tr>
<td>80 by 80</td>
<td>903</td>
<td>552</td>
<td>3.86</td>
</tr>
<tr>
<td>88 by 88</td>
<td></td>
<td>656</td>
<td>3.93</td>
</tr>
<tr>
<td>96 by 96</td>
<td></td>
<td>789</td>
<td>3.89</td>
</tr>
<tr>
<td>100 by 100</td>
<td>1431</td>
<td>847</td>
<td>3.94</td>
</tr>
</tbody>
</table>

NWB The number of boundary elements in the window.

NPB(I,J) The number of boundary elements in the picture that corresponds to the area covered by the window. NPB(I,J) changes as the window is moved to different I and J locations.
The mean value of the data in the window, \( W = \frac{\text{NW}B}{(\text{ROW})(\text{NCOL})} \).

The mean value of the data in the picture that corresponds to the area covered by the window, \( P(I, J) = \frac{\text{NPB}(I, J)}{(\text{ROW})(\text{NCOL})} \).

The numerator of the correlation coefficient, \( C(I, J) \), can then be written as:

\[
C(I, J) = K_1 \cdot \text{NCOR}(I, J) + K_2 \cdot [\text{NPB}(I, J) - \text{NCOR}(I, J)] \\
+ K_3 \cdot [\text{NW}B - \text{NCOR}(I, J)] + K_4 \cdot [\text{ROW} \cdot \text{NCOL} - \text{NPB}(I, J)] \\
+ \text{NCOR}(I, J) - \text{NW}B] ,
\]

where

\[
K_1 = (1 - W)(1 - P(I, J)) , \quad K_2 = -W(1 - P(I, J)) , \quad K_3 = -[1 - W]P(I, J)
\]

and

\[
K_4 = W \cdot P(I, J)
\]

Since \( 0 < W, P(I, J) < 1 \), then \( K_1, K_4 > 0 \) and \( K_2, K_3 < 0 \), and \( K_1 \) and \( K_4 \) are always greater than \( K_2 \) and \( K_3 \) in absolute magnitude. If the constants, \( K \), are thought of in terms of rewards and penalties, then the first term of equation (11) is a reward times the number of boundary elements that match in the window and picture. The second and third terms, since they are negative, penalize the mismatch of boundary and nonboundary elements in the picture and window, respectively. The last term is a reward for nonboundary elements that match in the picture and window. It is interesting to note that for \( W + P(I, J) < 1 \), then \( K_4 > K_4 > K_2, K_3 \); that for \( W + P(I, J) = 1 \), then \( K_4 = K_4 > K_2, K_3 \); and that for \( W + P(I, J) > 1 \), then \( K_4 > K_4 > K_2, K_3 \). This indicates that the binary correlation procedure gives a larger reward for matching up whatever occurs the least in the window or picture, boundaries or nonboundaries. This is also what a human observer would tend to do. To determine a normalization factor for equation (11), consider a perfect match for the window and picture; i.e., \( \text{NW}B = \text{NPB}(I, J) = \text{NCOR}(I, J) \) and \( W = P(I, J) < 1 \). Equation (11) becomes
\[ C(1,J) = (1 - W)^2 \cdot NWB + W^2 \cdot (NROW \cdot NCOL - NWB) \]

\[ = [1 - P(1,J)]^2 \cdot NPB(1,J) + [P(1,J)]^2 \]

\[ [NROW \cdot NCOL - NPB(1,J)] \tag{12} \]

Thus, for a nonperfect match, the normalization factor should be some combination of the two expressions in equation (12). Since the two expressions in equation (12) are the variances of the window and picture respectively, the correct normalization factor would be the square root of the products of the two expressions to produce a correlation coefficient.

It does not appear that computing the correlation coefficient would add a significant amount of running time to the binary correlation routine. This is because the mean value and variance of the window need only be calculated once and

\[ NWB = KMAX + \sum_{K=1}^{KMAX} NSQW(K) \]

The calculation that would require additional time and storage is determining NPB(1,J) for the mean value and variance of the picture for each shift of the window. In most cases, however, it is anticipated that it will be necessary to compute only NCOR(1,J).

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
 Marshall Space Flight Center, Alabama, November 19, 1973
APPENDIX A. BINARY CORRELATION LISTING

FORTRAN SOURCE LIST

5245. CAMPBELL, U.50145.30,

0 SUBITC MAIN
1 DIMENSION NW(200), FMT(12), MCOR(101, 101), ID(1324), JD(1324)
2 DIMENSION NSQM(1324), IP(3300), JP(3300), NSQP(3300)
3 READ 3, NROW, NCOL
4 300 FORMAT(15)
5 PRINT 302, NROW, NCOL
6 302 FORMAT(1X, SHNROW=P, 14, 1X, SHNCOL=P, 14, /)
7 READ(5, 301)(FMT(L), L=1, 12)

17 NROW = NROW + 1
18 NCOL = NCOL + 1
19 MCOR(NROW, NCOL+1) = 0
20 K = 0
21 DO 20 I = 1, NROW
22 IN = -1
23 READ(5, FMT)(MW(I), II=1, NCOL)
24 DO 10 J = 1, NCOL
25 NCU(J, J) = 0
26 IF(NW(I) = EQ. 0) GO TO 5
27 IF(IN GT 0) GO TO 3
28 K = K + 1
29 IN = I - 1
30 JW(K) = J - 1
31 YSQ(K) = 0
32 IN = 1
33 GO TO 10
34 YROW(K) = YROW + 1
35 GO TO 10
36 IN = -1
37 10 CONTINUE
38 KMAX = K
39 READ(5, 301)(FMT(L), L=1, 12)
40 YROW2 = YROW + YROW
41 YCOL2 = YCOL + YCOL
42 L = 0
43 DO 40 I = 1, NROW
44 IN = -1
45 READ(5, FMT)(NW(I), II=1, NCOL2)
46 DO 30 J = 1, NCOL2
47 IF(NW(J) = EQ. 0) GO TO 25
48 IF(IN GT 0) GO TO 23
49 L = L + 1
50 J = I - 1
51 IF(L GT NROW1) GO TO 30
52 IMAX = L
53 GO TO 20
54 J = I - 1
55 20 CONTINUE
56 NSUP(L) = 0
57 IL = I
58 JPL = J
59 IN = 1
60 IF(I, GT, NROW1) GO TO 30
61 LM = IMAX
62 GO TO 20
63 NSQP(L) = NSUP(L) + 1
64 GO TO 40
65 25 CONTINUE
66 23 CONTINUE
67 20 CONTINUE
68 10 CONTINUE
69 5 CONTINUE
FORTRAN SOURCE LIST MAIN

127 LMAX=L
130 KSTART=1
131 ISUM=0
132 CALL TIMNOW(IT1)
133 DO 100 L=1,LMAX
134 NJ=JP(L)+NSQP(L)
135 DO 80 K=1,KMAX
136 IF(JW(K).GE.IP(L))GO TO 100
141 I=IP(L)-1W(K)
142 IF(I.LE.1)GO TO 45
145 IF(NU.GT.JW(K))GO TO 46
150 GO TO 100
151 45 IF(NU.LE.JW(K))GO TO 80
154 46 JL=JP(L)-JW(K)-NSW(K)
155 IF(JL.GT.NCOL1)GO TO 80
160 IU=NU-JW(K)
161 IF(JL.LE.1JGO TO 76
164 IF(NSW(L).LE.NS(QW(K))))GO TO 53
167 MINL=NSQW(K)
170 GO TO 54
171 53 MINL=NSQP(L)
172 54 IF(MINL.GT.0)GO TO 60
175 IF(JU.LE.NCOL1)GO TO 55
200 JU=NCOL1
201 55 DO 56 J=JL,JU
202 56 ISUM=ISUM+1
203 GO TO 30
206 60 JM=JU-MINL
207 VAO=0
210 IF(JU.LE.NCOL1)GO TO 64
213 JU=NCOL1
214 64 DO 68 J=JL,JU
215 ISUM=ISUM+1
216 IF(J.GT.JM)GO TO 66
221 IF(NAD.GT.MINL)GO TO 68
224 VAO=NAD=1
225 GO TO 68
226 66 NAD=NU=1
227 68 NCOR(I,J)=NCOR(I,J)+NAD
231 GO TO 80
232 70 NSPAN=JW(K)+NSW(K)+1
233 IF(JP(L).GT.JW(K))GO TO 71
236 IF(NSPAN.GE.NU)GO TO 69
241 NAD=NAD+1
242 GO TO 75
243 69 VAO=JU
244 GO TO 76
245 71 IF(NSPAN.GE.NU)GO TO 72
250 NAD=NSPAN=JP(L)
251 IF(NSPAN,GT.NSUW(K))GO TO 74
254 MINL=NSQP(L)
255 GO TO 76
256 74 MINL=NSUW(K)
257 76 JM=JU-MINL
**FORTRAN SOURCE LIST: MAIN**

```
546, CAMPUSLL  140145.00;
15N  SOURCE STATEMENT
160   JL=1
200   GO TO 64
204   75  NAD=NSQP(L)+1
205   JN=JU=NAD+2
206   DO 78 J=1,JU
207   ISUM=ISUM+1
208   IF(J,LT,JM)GO TO 78
271   VAD=VAD+1
272   78  VCOR(I,J)=NCOR(I,J)+NAD
274   80  CONTINUE
276   100 CONTINUE
300   LMAX1=LMAX1+1
301   DO 200 L=LMAX1,LMAX
302   NU=JP(L)+NSQP(L)
303   DO 180 K=KSTR+1,KMAX
304   IF(I,L,I-W(K))GO TO 85
310   IF(NU.LE.JW(K))GO TO 180
313   JL=JP(L)-JW(K)-NSQP(K)
314   IF(JL.GT.NCOL1)GO TO 180
317   JU=NU-JW(K)
320   IF(JL.LT.1)GO TO 701
323   IF(NSQP(L).LE.NSW(K))GO TO 531
326   MINL=NSQP(K)
327   GO TO 541
330   531 MINL=NSQP(L)
334   IF(MINL.GT.0)GO TO 601
337   JU=NCOL1
340   552 NU JU=NCOL1GO TO 552
341   JU=NCOL1
346   GO TO 180
348   601 JM=JU-MINL
349   602 NAD=JU-JW(K)
350   603 IF(JU.LE.NCOL1)GO TO 641
352   JU=NCOL1
353   641 DO 661 J=JL,JU
354   ISUM=ISUM+1
355   IF(J.GT.JM)GO TO 661
360   IF(VAD.GT.MINL)GO TO 641
363   NAD=NAD+1
364   GO TO -91
365   661   NAD=NAD+1
366   661 NAD=NCOR(I,J)+NAD
370   GO TO 150
371   150 NSPAN=JW(K)+NSQP(K)+1
372   IF(JP(L).LT.JW(K))GO TO 711
375   IF(NSPAN.GE.JU)GO TO 691
400   NAD=NSQP(K)+1
401   GO TO 731
404   691 VAD=JU
405   711 IF(NSPAN.GE.NU)GO TO 731
407   NAD=NSPAN-JP(L)
```
IF(NSQP(IP),GT,NSQM(K))GO TO 741
MINL=NSQP(IP)
GO TO 761
741 MINL=NSQM(K)
761 JM=JU-MINL
JL=1
GO TO 641
731 NADV=NSQP(L)+1
751 JM=JU-NADV+2
DO 781 J=J1,JU
ISUM=ISUM+1
IF(JL.LT.JM)GO TO 781
NADV=NADV-1
781 NCOR(I,J)=NCOR(I,J)+NAD
180 CONTINUE
GO TO 200
85 ASTRTH=1
137 GO TO 52
200 CONTINUE
CALL TIMNO(IT2)
PRINT 5555,ISUM
5555 FORMAT(I5,THESUM=,F17.)
JTIME=IT2-IT1
PRINT 318
318 PRINT 999
319 FORMAT(5X,9HREGISTRATION COMPLETE)
37 JSTRTH=1
460 JSTOP=20
405 PRINT 403
403 FORMAT(1H1,5X,18HCORRELATION MATRIX)
401 DO 406 J=1,NRM1
402 PRINT 402,NCOR(II,J),J=JSTRTH,JSTOP
401 CONTINUE
474 IF(JSTOP.NQ.NCOL1)GO TO 406
477 JSTRTH=JSTOP+1
500 JSTOP=JSTOP+20
501 IF(JSTOP.LE.NCOL1)GO TO 405
504 JSTOP=JU1
95 GO TO 455
400 CONTINUE
STOP
END
APPENDIX B. CORRELATION ROUTINE LISTING

5245, CAMBELL 1140145.CO. FORTRAN SOURCE LIST

SOURCE STATEMENT
0 9IBF.C OR DATE
1 SUBROUTINE FORC(NP,MM,NCOR)
2 **********************************************************************
3 THIS SUBROUTINE CONTAINS THE STANDARD CORRELATION CALCULATION
4 **********************************************************************
5 DIMENSION NH(5,12),NP(112,112),NCOR(12),FNT(12)
6 COMMON/NAME1/ NCOL
7 COMMON/NAME2/ NROW
8 COMMON/NAME4/ FNT
9 READ (5) NN
10 FORMAT(2A6)
11 READ(5,FMT) (NN(I1),I=1,NCOL)
12 CONTINUE
13 READ(5,FMT) (NN(I1),I=1,NCOL)
14 CONTINUE
15 READ(5,FMT) (NP(I2),I=1,NCOL)
16 CONTINUE
17 READ(5,FMT) (NP(I2),I=1,NCOL)
18 CONTINUE
19 READ(5,FMT) (NP(I2),I=1,NCOL)
20 CONTINUE
21 READ(5,FMT) (NP(I2),I=1,NCOL)
22 CONTINUE
23 READ(5,FMT) (NP(I2),I=1,NCOL)
24 CONTINUE
25 READ(5,FMT) (NP(I2),I=1,NCOL)
26 CONTINUE
27 READ(5,FMT) (NP(I2),I=1,NCOL)
28 CONTINUE
29 READ(5,FMT) (NP(I2),I=1,NCOL)
30 CONTINUE
31 READ(5,FMT) (NP(I2),I=1,NCOL)
32 CONTINUE
33 READ(5,FMT) (NP(I2),I=1,NCOL)
34 CONTINUE
35 READ(5,FMT) (NP(I2),I=1,NCOL)
36 CONTINUE
37 READ(5,FMT) (NP(I2),I=1,NCOL)
38 CONTINUE
39 READ(5,FMT) (NP(I2),I=1,NCOL)
40 CONTINUE
41 READ(5,FMT) (NP(I2),I=1,NCOL)
42 CONTINUE
43 READ(5,FMT) (NP(I2),I=1,NCOL)
44 CONTINUE
45 READ(5,FMT) (NP(I2),I=1,NCOL)
46 CONTINUE
47 READ(5,FMT) (NP(I2),I=1,NCOL)
48 CONTINUE
49 READ(5,FMT) (NP(I2),I=1,NCOL)
50 CONTINUE
51 READ(5,FMT) (NP(I2),I=1,NCOL)
52 CONTINUE
53 READ(5,FMT) (NP(I2),I=1,NCOL)
54 CONTINUE
55 READ(5,FMT) (NP(I2),I=1,NCOL)
56 CONTINUE
57 READ(5,FMT) (NP(I2),I=1,NCOL)
58 CONTINUE
59 READ(5,FMT) (NP(I2),I=1,NCOL)
60 CONTINUE
61 READ(5,FMT) (NP(I2),I=1,NCOL)
62 CONTINUE
63 READ(5,FMT) (NP(I2),I=1,NCOL)
64 CONTINUE
65 READ(5,FMT) (NP(I2),I=1,NCOL)
66 CONTINUE
67 READ(5,FMT) (NP(I2),I=1,NCOL)
68 CONTINUE
69 READ(5,FMT) (NP(I2),I=1,NCOL)
70 CONTINUE
71 READ(5,FMT) (NP(I2),I=1,NCOL)
72 CONTINUE
73 READ(5,FMT) (NP(I2),I=1,NCOL)
74 CONTINUE
75 READ(5,FMT) (NP(I2),I=1,NCOL)
76 CONTINUE
77 READ(5,FMT) (NP(I2),I=1,NCOL)
78 CONTINUE
79 READ(5,FMT) (NP(I2),I=1,NCOL)
80 CONTINUE
81 READ(5,FMT) (NP(I2),I=1,NCOL)
82 CONTINUE
83 READ(5,FMT) (NP(I2),I=1,NCOL)
84 CONTINUE
85 READ(5,FMT) (NP(I2),I=1,NCOL)
86 CONTINUE
87 READ(5,FMT) (NP(I2),I=1,NCOL)
88 CONTINUE
89 READ(5,FMT) (NP(I2),I=1,NCOL)
90 CONTINUE
91 READ(5,FMT) (NP(I2),I=1,NCOL)
92 CONTINUE
93 READ(5,FMT) (NP(I2),I=1,NCOL)
94 CONTINUE
95 READ(5,FMT) (NP(I2),I=1,NCOL)
96 CONTINUE
97 READ(5,FMT) (NP(I2),I=1,NCOL)
98 CONTINUE
99 READ(5,FMT) (NP(I2),I=1,NCOL)
100 CONTINUE
101 CONTINUE
102 RETURN

END
APPENDIX C. FAST FOURIER TRANSFORM CORRELATION LISTING

5245, CAMPBELL +140145.00

FORTRAN SOURCE LIST

SOURCE STATEMENT

0     01BFTC MAIN
1     DIMENSION XNPR(64,64), XNMI(64,64), XNPR(64,64), XNPI(64,64)
2     DIMENSION IA(64,64)
3     READ 1,NROW,NCOL
4     FORMAT(2IS)
5     CALL IN(XNPR,XMPI,NROW,NCOL)
6     NTOT=NROW*NCOL
7     CALL TIMMD(11)
8     CALL FFT1(XNPR,XMPI,NTOT,NROW,NCOL,1)
9     CALL FFT1(XNPR,XMPI,NTOT,NROW,NCOL,1)
10    CALL FFT1(XNPR,XMPI,NTOT,NROW,NCOL,1)
11    CALL FFT1(XNPR,XMPI,NTOT,NROW,NCOL,1)
12    CALL FFT1(XNPR,XMPI,NTOT,NROW,NCOL,1)
13    XNMTOT=NTOT
14    DO 7 J=1,NROW
15    DO 6 I=1,NCOL
16    STORE=XNPR(1,J)+XNPR(I,1)\+XNPI(1,J)*XNPI(I,1)
17    XNPR(1,J)=XNPI(1,J)*XNMI(I,1)*XNNII(I,1)*XNPR(1,J)/XNMTOT
18    XNPR(I,J)=STORE/XNMTOT
19    CALL FOPEN(11,11)
20    CALL FOPEN(11,11)
21    CALL FOPEN(11,11)
22    CALL FOPEN(11,11)
23    CALL FOPEN(11,11)
24    CALL FOPEN(11,11)
25    CALL FOPEN(11,11)
26    CALL FOPEN(11,11)
27    CALL FOPEN(11,11)
28    CALL FOPEN(11,11)
29    CALL FOPEN(11,11)
30    CALL FOPEN(11,11)
31    CALL FOPEN(11,11)
32    CALL FOPEN(11,11)
33    CALL FOPEN(11,11)
34    CALL FOPEN(11,11)
35    CALL FOPEN(11,11)
36    CALL FOPEN(11,11)
37    CALL FOPEN(11,11)
38    CALL FOPEN(11,11)
39    CALL FOPEN(11,11)
40    CALL FOPEN(11,11)
41    CALL FOPEN(11,11)
42    CALL FOPEN(11,11)
43    STOP
44    END
REFERENCES


