U.S. DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

# A 2D FFT filtering program for image processing with examples

by

Ken Watson [1]

Open-File Report 92-265

1992

DISCLAIMER

The program in Appendix A was written in Fortran-77 for a SUN 3/470 computer[2]. Although program tests have been made, no guarantee (expressed or implied) is made by the author regarding program correctness, accuracy, or proper execution on all computer systems. This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards.

[1] Denver, Colorado.

[2] Any use of trade names in this report is for descriptive purposes only and does not imply endorsement by the U.S. Geological Survey.

CONTENTS

Abstract . . . . . . . . . . 1

Introduction . . . . . . . . . . 1

Method . . . . . . . . . . 2

Simple noise patterns . . . . . . . . . . 4

Filter design . . . . . . . . . . 6

Application to noise removal in aircraft and satellite data . . . . . . . . . . 7

Filtering strategies . . . . . . . . . . 9

Additional uses of the 2D FFT . . . . . . . . . . 10

Summary . . . . . . . . . . 12

Bibliography: . . . . . . . . . . 12

APPENDIX A. **Fft2d.filter program** . . . . . . . . . . 14

APPENDIX B. 2D FFT of some fundamental patterns. . . . . . . . . . . 31
- Image impulse. . . . . . . . . . . 31
- Periodic pattern of impulses along edge. . . . . . . . . . . 31
- Image "line". . . . . . . . . . . 32
- Periodic line pattern. . . . . . . . . . . 32
- Evenly spaced point pattern. . . . . . . . . . . 33

# A 2D FFT filtering program for image processing with examples

Ken Watson
Branch of Geophysics, U.S. Geological Survey
Box 25046 MS 964, Denver CO 80225

**Abstract**

The 2D FFT is a powerful means for removing noise from images because efficient filters can be designed based on the observed spatial or frequency patterns of the noise. This paper provides a generalized computer program for filtering images together with an illustrated introduction to the main concepts of the 2D Fast Fourier Transform in image processing and examples of noise removal from a variety of spacecraft and aircraft systems. Following the examples a general strategy for filtering is developed: filter only derivative products (not original images), first remove noise evident in the transform, and apply a "minimum" filter to reduce residual noise.

In addition to noise filtering the method also has application to such diverse image processing problems as enlargement, instrument response correction, registration, and extraction of albedo and slope information. Examples are provided to illustrate these concepts.

**Introduction**

(The text that follows is an expanded version of a paper submitted to Geophysics and in addition includes a generalized filtering program in an appendix.) Noise is often present in remote sensing images and particularly data acquired from aircraft. Although not visually evident in the original images, noise can be a significant problem in computed derivative images (ratios, principal components, thermal inertia, band depth, or other analytical algorithms). There are several advantages to using a two-dimensional Fast Fourier Transform(2D FFT) to reduce noise by filtering in the frequency domain. Coherent noise is generally conspicuous in an image display of the amplitude of the Fourier transform either as bright spots or lines. When noise is observed as a pattern of harmonics (e.g., due to an AC power supply), the mathematical relation of the harmonics can be used to design a noise removal filter that extends to other frequencies where the noise is not as obvious. Sometimes, noise in an image is not immediately apparent in the transform. If the pattern in the spatial domain has a simple form, then a mathematical relationship can be used to relate the noise pattern on the image to its transform. The development of a filter involves experimenting with methods to block out, to interpolate across, or to reduce smoothly the observed bright pattern. Standard image processing methods (i.e. contrast stretching and enlarging) can then be applied to enhance this area of the transform image that included the bright pattern and find local maxima for constructing an appropriate filter. In addition, the FFT approach provides a method to model noise (Rose, 1989) and, in some cases, to detect the source of the noise and to develop special purpose filters for similar data sets.

The fast Cooley-Tukey algorithm for computing discrete Fourier transforms provides a means for filtering in the frequency domain and has greatly expanding the field of digital signal processing (Deregowski, 1971). Recent developments in array processors for parallel execution of data, increased machine speed using reduced instruction set computers (RISC), and faster CPUs have removed the speed limitation of the 2D FFT. It can now be used as a tool for noise removal on systems ranging from mainframes to personal computers (PCs).

Only certain types of noise are suitable for treatment using the FFT method, in particular, stationary periodic noise that produces repetitive patterns in the image. Random noise, which appears as speckle patterns, or individual scanline dropouts, require specialized algorithms. Because the common sources of noise are power supplies, recording systems, and banks of detectors, most noise has a periodic component. The requirement that noise be stationary implies that repetitive noise is consistent throughout the image. Non-stationary noise ( i.e., noise that varies in frequency throughout the image) is difficult to remove and linear filtering techniques are not generally effective.

In addition to noise removal, the 2D FFT method can also be applied to such diverse problems as enlargement, instrument response correction, image registration, and extraction of albedo and slope information. Image enlargement which employs extending the high frequencies by filling the transform to the required size using zeroes is the least biased of all enlargement methods and is particularly effective for high enlargement factors. The instrument response function can be derived by viewing an illuminated known target because the resulting response is the convolution of that function with the scene radiance. Geometric registration between images of comparable appearance can be done using cross correlation of image segments. (Because correlation and convolution can be expressed as multiplications in Fourier transform space they can be computed using the 2D FFT.) Albedo and slope information often appear to be associated with low and high frequencies respectively (Eliason et al., 1981) thus spatial filtering can be applied to examine or enhance this information.

One of the reasons that the 2D FFT method has been used infrequently in the remote sensing community has been it's complexity. This paper provides a comprehensive computer program to apply filters, examples of their use, and a simple introduction to the concepts of filtering in the frequency domain. Some caveats about the design of filters are provided including a method to avoid introduction of unacceptable artifacts. Examples of the use of the 2D FFT for image enlargement, geometric registration and extraction of spatial frequency information are presented to illustrate the general versatility of the method. Readers who wish to pursue the 2D FFT in greater detail are advised to consult the appropriate literature (Bracewell,1986; Clement, 1973; Gillespie, 1980; Oppenheim and Schafer, 1975).

**Method**

Because the purpose of this paper is to describe the application of 2D FFT filtering to images, it is necessary to distinguish among images, their transforms, and image representations of transforms henceforth referred to as spatial images, transforms, and frequency images. An image can be considered as a matrix of integer values. Let the spatial image I(j,k) be an integer array j=0,N-1; k=0,N-1. (A square image is used to keep the initial expressions simple. However, the filtering program is designed for rectangular images). Then the 2D Fourier transform (Pratt, 1978) can be mathematically defined in spatial frequency space (u,v) as:

$$T(u,v)=\frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} I(j,k)e^{-\frac{2\pi i}{N}(uj+vk)};\ i=\sqrt{-1}. \tag{1}$$

The inverse transform is given by

$$I(j,k)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)e^{\frac{2\pi i}{N}(uj+vk)}. \tag{2}$$

Note that although the spatial image is a set of integers, its transform is both complex (i.e. consisting of a real and an imaginary part) and non-integer. (Because the real part is even and the imaginary part is odd this is referred to as conjugste symmetry.) In order to display the transform as a frequency image, an output form must be selected (i.e. modulus or amplitude, phase, real or imaginary part). The values are then rescaled (generally between 0-255) and truncated to integers for display. An image is filtered by computing the transform, applying a filter function to both the real and imaginary parts, forming the inverse transform, and then rounding to integers. In the filter program (Appendix A), a standard but well-known and versatile method is used for computing the 2D transform. It is the Cooley-Tukey Fast Fourier Transform that has been available in a number of forms for many years and is, thus, either commonly available (Press et al., 1986) or easily coded for any machine.

The program call is a single line in the filter program :

```
        call fourt (data,nn,ndim,s,1,work)
```

where s=-1 for the forward transform and +1 for the inverse transform. The image(s) are stored in common in a complex array called data. The remaining variables in the call statement are: nn, a two dimensional vector specifying the image size; ndim=2, the number of dimensions of the transform; and work, a work array. There is complete generality in the size of the image (subject to the computer memory) although the algorithm is fastest when the number of elements per line (pixels) and the number of lines are rich in prime numbers (Singleton, 1969). Use of images whose dimensions are powers of two is optimal when using the FFT algorithm (Singleton, 1967).

There are certain, well-known properties of the transform that are used in the filter program (linearity, symmetry, evenness). Proofs can be found in standard texts (e.g. Bracewell, 1986; Oppenheim and Schafer, 1975; Brigham, 1974) and are not presented here. Because of the additive properties of the transform, a pair of images can be filtered simultaneously by placing one image into the realpart and the other image into the imaginary part of the complex array data. The filtered pair are then extracted from the real and imaginary parts of the inverse of the filtered transform. (Although this result may sound counterintuitive to some readers it is a consequence of the conjugate symmetry mentioned previously. The real part of the transform of a real function is even and the imaginary part is an odd function, whereas the real part of the transform of an imaginary function is odd and the imaginary part is even. Thus the contributions of the two input images are in fact separable in the transform.) This property saves processing time when applying the same filters to multiband images such as the Thermal Infrared Multispectral Scanner (TIMS), the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), or the Landsat Thematic Mapper (TM).

The filter program (Appendix A) has been written to process a maximum of 6 images with each set of filters. Modification to a larger number (it is best to work with pairs) requires changing only the image management files. The disk input/output (i/o) controls follow the standards of the USGS image processing software REMAPP (Sawatzky, 1985) and its PC form (Livo, 1990). Implementation of this program with other image processing programs will require some understanding of the basic i/o calls that are required. The changes are not very substantial as the image(s) are input as a block, transformed, filtered, and output as a block. There is also an option to output a scaled image of the transform before or after filtering.

A transform can be displayed in two ways. The standard method is to display directly the output from the transform algorithm and that will be the method employed in this paper. In that case the dc term(zero frequency, average value of image) occurs at pixel 1, line 1 (fig. 1, left). (The folding or Nyquist frequencies of the transform are at the mid points of the pixel and line axes). There is another display method, called the optical transform, which interchanges the quadrants and places the dc term (and thus the center of the frequency coordinate axes) in the center of the image (fig. 1, right) thus making the form of the symmetry more obvious. The optical transform involves reshuffling the output from the 2D FFT and is unnecessary for filtering; therefore, it is primarily used only as an instructional tool. Because an image consists of a set of positive integers (fig. 2A), then its transform is conjugate symmetric and the image of the transform (frequency image) is symmetric (fig. 2B). (The symmetry is more evident later in complex patterns such as fig. 21D)

There are several aspects of the frequency image worth noting. The dc term or average value of the image appears as a spike in the transform. When the frequency image is re-scaled to enhance subtle features, the dc term is excluded from the scaling because its value is much greater than the rest of the transform and its inclusion would excessively compress rescaling of the remaining values. Instead the dc term is set to the maximum display value (generally 255). For display purposes, options are provided in the computer program (Appendix A) to output either the transform or the log[transform]. There are advantages to both options but generally the log[transform] is most useful for detecting noise. Because the transform is symmetric, filters are applied that are symmetric and the same filters are used for both the real and imaginary parts of the transform in order to avoid phase distortion.

An important concept of the Fourier transform (one sometimes described as a mathematical representation of the uncertainty principle called the similarity theorem) is that features which occupy only a limited part of the spatial domain are spread out in the frequency domain and *vice-versa*. Consequently a noise pattern that appears to be present throughout the spatial image will occupy a restricted area of the transform and can be isolated. A noise pattern that is present in only part of the spatial image will be spread out in the transform and difficult to isolate. This concept will be more evident in the examples. (A reassuring exercise is to use the filter program to demonstrate that the inverse transform of a transformed image generates the original image.)

The concept of transform pairs is embodied in the relationships between spatial image patterns and their frequency transforms, which can be viewed as equivalent.. In the following section, several examples of image transform pairs will be provided. The transform of noise patterns which can be identified in either the spatial image or the frequency image display can thus be determined.

**Simple noise patterns**

Coherent noise is generally quite evident in the image of the a transform because it is confined and thus appears anomalously bright. Consider a square image (N pixels, N lines) that consists of a single bright pixel at pixel P and Line L: an image impulse function (fig. 2A). The real part of its transform (fig. 2B) consists of a sinusoidal variation (somewhat like a corrugated roof). As the image and its transform are interchangeable then the transform of a sinusoidal spatial variation is an impulse in the frequency domain. The geometry of this relationship is shown on figure 3. The distance (D) between adjacent patterns (spatial periodicity) and the pattern slope ($\theta$) are illustrated for positive and negative slopes and the equations are provided below. The equations are provided below and the derivation is given in Appendix B).

Let $W = N/D$. Then

$L = 1 + W \cos\theta$ ;

$P = 1 + W \sin\theta$ if $\theta \geq 0$ and

$= N - W \sin|\theta|$ if $\theta < 0$ . (3)

In general for an N*M image

$W = \{N \sin|\theta| + M \mathrm{co}(\theta\}/[D\{\sin|\theta| + \cos\theta\}]$.

These results can be used to examine some simple relationships involving the position of the impulse, the image size, and extension to line patterns (Table One). The mathematical derivation beyond the single point cases can be constructed using superposition (Appendix Two). The paired figures accompanying Table One show an image of the impulse function (described in the table) and the real part of its transform.

Table One. List of transform pair examples

| Figure | Image size | | Pixels and lines of Impulse function | | Range |
|---|---|---|---|---|---|
| | N | M | P | L | |
| 4A,B | 64 | 64 | 1 | 4 | |
| 4C,D | 64 | 64 | 4 | 1 | |
| 5A,B | 64 | 64 | 3 | 4 | |
| 5C,D | 64 | 64 | 2 | 4 | |
| 6A,B | 64 | 128 | 2 | 4 | |
| 6C,D | 128 | 64 | 2 | 4 | |
| 6E,F | 128 | 64 | 2 | 3 | |
| 7A,B | 128 | 128 | 9 | 12 | |
| 7C,D | 128 | 128 | 1 | 1+I*16 | I=0,7 |
| 8A,B | 128 | 128 | I*10 | I*10 | I=1,12 |
| 8C,D | 128 | 128 | I*5 | I*10 | I=1,25 |
| 9A,B | 128 | 128 | I | 17 | I=1,128 |
| 9C,D | 128 | 128 | I | 1+J*16 | I=1,128;J=0,7 |

Figure 4 shows how the position of the impulse function changes to produce horizontal or vertical patterns. An impulse on the vertical (line) axis (fig. 4A) causes a horizontal pattern in the transform (fig. 4B). Conversely, from this result it can be seen that the transform of a horizontal scanline noise (Fig. 4) will be clustered along the line (vertical) axis of the transform (fig. 4A). As the spike moves away from the axes of the figure (fig. 5A and 5C) the resulting transform pattern (fig. 5B and 5.D) is rotated by an angle (slope) that depends on the position of the impulse spike (equation 3). Figure 6 demonstrates that the size of the image affects both the spacing and orientation of the transform pattern. Conversely, this implies that the position of a noise spike in transform space will change as the size of the image being transformed is changed.

The variation in the position of the noise spike can be easily deduced from equation (3) using figure 3 as an illustration. Consider two images of different sizes $(N_1,M_1)$ and $(N_2,M_2)$ which contain the same pattern. Because $\theta$ and D are the same then it follows that

$$(L_1 - 1)/W_1 = (L_2-1)/W_2$$
$$\text{and } (P_1-1)/W_1 = (P_2 -1)/W_2 \quad \text{if } \theta \geq 0$$
$$(N_1 -P_1)/W_1 = (N_2 - P_2)/W_2 \text{ if } \theta < 0 \tag{4}$$
where $W_1/W_2 = \{N_1 \sin|\theta| + M_1 \cos\theta\}/\{N_2 \sin|\theta| + M_2 \cos\theta\}$.

(This result could also have been determined by proportionate scaling of the position of the noise spike.) In the example shown, figure 6, $N_1=64$, $M_1=128$, $N_2=128$, $M_2=64$; $P_1=2$, $L_1=4$. To preserve the same pattern in figure 6B then from equation (3) $\tan\theta = 1/3$ and from equation (4) $W_1/W_2=1.4$ and $P_2 \sim 1.7$, $L_2 \sim 3.1$. Because only integral pixel and line values can occur, the pattern is approximated by using a transform of an impulse at $P=2$, $L=3$. The result is shown in figure 6E and F. This result shows that the transform pattern is dependent on the image size. Often to save computer time it is desirable to work with image windows whose dimensions are powers of two. In many cases the images may not be so conveniently sized. After the noise has been identified using an optimum window size the noise must be removed from a different size image. It is thus important to recognize how the noise changes with changing image size.

Figure 7 illustrates the effects of periodicity. A single impulse (fig. 7A,B) has a corrugated (sinusoidal) transform whereas a periodic (harmonic) impulse pattern has a transform that appears as a periodic, sharply defined bright lines like a row of "walls". This transform is an illustration of the uncertainty principle concept mentioned in the previous section and can be visualized as a compression of the corrugated pattern in figure 7B. Conversely the corresponding transform of these patterns is "spread out" as represented in the change from single to periodic impulses (fig. 7A,C).

Figure 8 extends the previous results to an inclined harmonic pattern. Note that the pattern in figure 8C must wrap around as a consequence of the assumption that the image is periodic in both directions. This geometry can be visualized by regarding the image as a tile with the periodic pattern present in a mosaic of tiles. Figure 9 illustrates the transform relationships for line patterns and can be compared with the equivalent results for a point in figures 4A,B and 7C,D. The comparison illustrates two important concepts. It reiterates the uncertainty principle concept that as the pattern expands from a point to a line pattern (or a periodic point to a periodic line pattern) the resulting transform pattern is compressed. It can also be deduced (fig. 9C,D) that the periodic point and periodic line patterns are transform pairs (Appendix B). Scanline noise is a common problem in remote sensing images and this result shows that the transform of a horizontal line noise should appear as spikes along the line axis. The position of the noise along the transform axis (frequency) is dependent on the periodic spacing in the spatial image.

## Filter design

Removal of noise from an image first requires identification of the noise either in the spatial image or the frequency image. Once the noise pattern is identified, the computer program (Appendix A) allows for three methods of noise suppression: blocking, smoothing, and interpolation. The blocking filter sets values of the transform to zero. The smoothing filter is an extension of the blocking filter and uses a rounding algorithm based on the sinc function (Bracewell, 1986) to taper values at the edges of the filter. The interpolation filter uses values just beyond the edge of the noise and bilinearly interpolates across the noise pattern. This latter filter represents the least intrusive filter and is useful for testing whether a blocking or smoothing filter is necessary.

Selection of an appropriate filter is dependent upon the type of noise present. Several geometric filter shapes are provided for various types of noise patterns: point, full line, rectangular, periodic block, wedge, angular harmonic, and "U-shaped". The "U-shaped" filter is a two dimensional filter can be visualized as a bathtub that has been cut in half across its width. It is used to reduce noise that clusters along either the line or pixel axis in the frequency domain. In addition a low and a high pass filter are provided for general image enhancement. Both these filters use a rapid roll off/on in the vicinity of a specified radius from the dc term. The low pass filter sets high frequency values beyond this radius to zero. The high pass filter reduces low frequency values inside the specified radius and increases high frequency values beyond it.

A brief discussion of the need for a smoothing filter is necessary. In some cases a blocking filter may causes artifacts (ringing) due to its sharpness. Based on our previous discussion of the uncertainty principle we expect that a filter which is narrowly confined in frequency space will have a transform that will be spread out in image space. Conversely if a filter is allowed to spread out smoothly then its transform will be more confined. A filter is applied to an image transform by multiplication in transform space. When the inverse transform is applied to recover the filtered image, the process is mathematically equivalent to convolving the original image with the inverse transform of the filter. If the transform of the filter is "spread out", then there is greater likelihood of producing undesirable artifacts. The smoothing filter represents a compromise between a blocking filter, which is sharply confined in the frequency domain, and a filter that occupies the entire transform space.

**Application to noise removal in aircraft and satellite data**

The most common noise problem in remote sensing images is scanline noise. A single channel of a commercial aircraft imaging spectrometer (GERIS) of an area near Cripple Creek, CO shows a periodic scan line noise (fig. 10A). The noise is evident on every fourth horizontal line and, using equation (3) ($M=400$, $\theta=0$, $D=4$), we can compute the position of the transform of this noise at $P=1$, $L=101$. This can be seen as a spike in the transform image (fig. 10B). A point blocking filter was then applied and the results are shown (fig. 11A). There is some leakage of the noise beyond the spike as seen in the filtered image and so an extended filter was applied at $L=101$ for 3 pixels on either side of the spike (i.e. $P=1,4$ and $P=510,512$). This filter (fig 11B) provides a satisfactory removal of the scanline noise.

Frequently noise that is not evident in the original image can be seen in derivative (specially processed) images. A TM satellite band ratio image (fig. 12A) of Bolivia exhibits periodic noise which appears as a series of bright spikes in the transform due to multiple harmonics (fig. 12C). A blocking, periodic box filter (together with some supplementary filters to be discussed later) was applied (fig. 12D). The transform appears black in those areas where the filter was applied and the filtered image shows considerable reduction in scanline noise (fig. 12B).

Often in aircraft data, the scanline noise is not substantially periodic. An example is shown using the second principle component of TIMS data from Iron Hill, CO (fig. 13A). The scanline noise appears in the transform as bright areas along the vertical edges(fig. 13C). The 'U-shaped' filter, which is useful in this case because the pattern is not entirely periodic, requires experimenting with the cutoff limits of the filter. A dark band along the vertical edges of the transform (fig. 13D) shows the location of the filter. The resulting filtered image (fig. 13B) can be compared with the original (fig. 13A) to see the improvement. (The difference between these two images is shown in figure 13E to illustrate the noise that was removed.)

Multiple noise patterns are commonly present. In some cases, this type of noise is evident in the transform as isolated bright spots or lines (e.g. fig. 13C) and in others the patterns appear to be repetitive (harmonic). A second principal component of TIMS aircraft data (fig. 14A,C), this time for Canon City, Colorado, shows a much more extensive noise pattern (fig. 14C) then in the Iron Hill example (fig. 13A,C). For this image several blocking filters were employed including: U-shaped, full line and angular harmonic (see fig. 14D). The improvement in the filtered image is evident (fig. 14B). The filters can be seen by comparing figure 14C with the dark areas in figure 14D. The 'U-shaped' filter, which is only one pixel wide, is barely noticeable and only in the upper left-hand corner. The full line filters appear as dark lines. The angular harmonic filter appears as a set of short vertical line segments starting in the lower left and (due to symmetry) upper right of the transform.

Sometimes, after the application of filters to more obvious transform noise patterns, a residual noise is revealed that was not initially evident. In the example images that follow (figs. 15-21), 6 panels labeled A through F are shown. The top panels A,B,and C are images and the bottom panels D,E, and F are their corresponding transforms. Panel A shows the starting image, panel B the filtered image and C the difference between A and B. To make comparisons easier, the results from a previous filtering step are repeated in the following illustration and an arrow is used to indicate the primary orientation of the noise pattern being filtered. Thus, panels A and D of figure 17 are identical to panels B and E of figure 16, and so forth.

A second principle component of TIMS aircraft data of De Weese Plateau, Colorado, is used to demonstrate multiple noise patterns that are superposed. The original image (fig. 15A) shows an intense scanline (horizontal) noise that is present in the transform along the edge of the vertical axis (fig. 15D). This noise is reduced (fig. 15B) using a 'U-shaped' filter (fig. 15E and F) along the edge. The filtered transform (fig. 15E and 16D) shows a series of bright full line (vertical) noise patterns. Using this filtered transform as a guide, it is relatively easy to locate and apply full line blocking filters (fig. 16E and 16F). The noise removed (fig. 16C) involves quite complex patterns and is not apparent in a side-by-side comparison of the image (fig. 16A) and its filtered counterpart (fig. 16B). The primary shape in the removed noise pattern is a sinusoidal variation in the horizontal direction repeated 11 times and is, thus, associated with the noise removed by the full line blocking filter that was applied at pixels 11 and 12.

A harmonic pattern is also present in the transform (fig. 16D and E) and appears as a set of short bright vertical lines starting in the lower left of the transform and progressing diagonally upward. Only a few harmonics are evident in the figure but more detailed examination, using an interactive computer image display to enlarge the transform, shows that this pattern continues for 7 harmonics. The pixel and line value for the center position of each harmonic is determined by finding the local maximum in the transform. The angle and frequency of the harmonic pattern can be computed by least squares fitting the pixel and line values to a harmonic function and, for this example, the computed angle and frequency were determined to be 5.17° and 84.34, respectively. Using these values an angular harmonic filter was applied to remove the noise. The filter can be seen as a series of black line segments in the filtered transform (fig 17E), and as bright line segments in the transform difference image (fig 17F). Comparison of figures 17 A and B shows that the pattern removed by this filter is a high frequency diagonal line banding (fig. 17E).

The angular harmonic filter can be determined directly from the noise pattern using equation (3) and the parameters described in figure 3. The line banding noise in figure 17A can be estimated by measuring the angle ($\theta \sim -10°$) and the spacing between the noise ($D \sim 100$). From equation (3), using $N=512$ and $M=1024$, then W can be computed as approximately 9.5 and thus $P=510$ and $L=10$. Because the noise pattern is more like a line than a corrugation (compare fig. 7B and 7D), there are higher frequency components present and, thus, a repetitive (harmonic) noise pattern can be observed in the transform. The noise should be of the form:

$$P \sim 512 - 9.i\sin 10° \qquad i=1,2,3,....$$
$$L \sim 1 + 9.5i\cos 10°$$

and the resulting {P,L} harmonic pairs are {510,10}, {509,20}, {507,29}, {505,48}, ... This, in fact, appears to coincide with the noise pattern observed in the upper right hand corner of the transform (fig. 17D). Because of the transform symmetry (see fig. 1, left) the symmetrical pair to {P,L] : {P',L'} is given by $P'=N+2-P$ and $L'=M+2-L$. It follows then that the symmetrical equivalents of the harmonic pairs can be written as: {4,1016}, {5,1006}, {7,997}, {9,978}, ... and these correspond to the areas in the lower left hand corner of the transform (fig. 15D) that were first identified as the repetitive noise pattern.

Although we appear to have removed all the noise patterns evident in the transform as bright areas, the resulting image (fig. 17B and 18A) still shows a pronounced diagonal pattern (see arrow) of dark, irregular lines. Using the technique just described the characteristics of this pattern: $\theta \sim 39°$ and $D \sim 60$ can be used to compute the approximate location of the noise in transform space at $P \sim 9$ and $L \sim 11$. Examination of the transform on a video display monitor shows a bright line segment centered at pixel 9, line 12 and extends from line 8 to 28 (not evident on figure 18D). A line segment blocking filter was applied (fig. 18E and F) and the resulting image (fig. 18B) and the companion difference image (fig. 18C) show the removal of this noise pattern.

The image appears at this stage to be fairly noise free, but careful examination reveals additional noise components. A second harmonic noise pattern was identified close to the vertical axis of the transform and filtered (fig. 19). A low frequency horizontal banding was also identified and removed, together with several short line segments of noise that appeared to cause some high frequency noise subparallel to the scanlines (fig. 20C).

Figure 21 is a summary of the complete filtering process. Panels A, B and C show respectively the original image (same as figure 15A), the completely filtered image (same as figure 20B), and the removed noise. Panels D, E, and F are the respective transforms and the location of the various filters can be seen. This example illustrates the power of the 2D FFT for producing a usable image from one with considerable noise.

**Filtering strategies**

From the previous examples, it is possible to outline a general procedure for filtering images. Often noise is not apparent in the original images, but only in derivative products (e.g. ratios, principle components). Filters should be applied only to these derivative products rather than the original images from which they were constructed. Noise which is evident in the transform as bright line segments should be filtered first before determining if more subtle noise is present. Although the intent of a filter is to remove only noise, some signal is also removed (and in some cases distorted), consequently it is desirable to apply only the "minimum filter" necessary. How does one determine this minimum filter and what are the consequences of applying a filter that is too strong or too extensive?

There is a common answer to both questions. Examination of the filtered image together with the difference image can indicate what the filter is doing and whether it is necessary. Using the principle of the minimum necessary filter, it is possible to fine-tune both the type and extent of the filter. Thus a small blocking filter is more desirable than a large one (if the resulting image is satisfactorily filtered), but a blocking filter is less desirable (since it removes all signal and noise at that part of the transform) than a multiplicative (smoothing) filter, which only reduces the signal and noise.

There is a second and potentially more important aspect of the multiplicative filter. When examples of transforms of simple patterns were discussed, the concept of the uncertainty principle was invoked to explain that a narrowly confined pattern produced an extended transform pattern and *vice versa*. In the same fashion, if a filter has very sharp cutoffs (e.g. blocking filter), then its effect in the image domain is spread out (unconfined). For example, a very sharp high frequency filter can cause a ringing artifact throughout the image. Although in figure 18B we can see that an unwanted noise pattern (fig. 18B) is being removed without introducing obvious artifacts the example in figure 20B is not as convincing. An examination of the effects of a filter require careful examination using a high-resolution image display screen. Depending on the purposes to which the images are to be employed, there are cases when it would be better not to apply sharp cutoff filters because the ringing can lead to a spurious "structural grain" in the image.

One final but important, issue has not been discussed. Some images are very large and it is impractical to apply the FFT method to the entire image due to processing time, available memory, or both. Thus, it would be convenient to filter smaller parts of the image and then reassemble them. If information has been filtered out then it is conceivable that the parts will not match. An example is shown in figure 22 using TIMS data of the De Weese Plateau area, Colorado. The top and bottom halves of the image (fig. 22A) were filtered using the same 'U-shaped' filter and the results appended back together (fig. 22B). The filtered image segments no longer match along the join line. The problem can be reduced, however, by applying the filter to overlapping segments and deleting the overlap (fig. 22C).

**Additional uses of the 2D FFT**

The presence of high frequency spatial information in an image is necessary in order for it to appear in focus. The absence of high frequencies is generally why some images (e.g. gravity data) often look fuzzy at large scales, as can be illustrated by applying a low pass filter to SPOT data of an area south of the town of Texas Creek, Freemont County, Colorado (fig. 23). Although this filter can be used to remove high frequency noise or to examine the low frequency content of an image, it does reduce the image sharpness. Conversely, a high pass enhancement filter (which increases high frequencies relative to low) can be used to sharpen a fuzzy image. Great care must be taken not to overdo the high frequency enhancement as noise patterns will also be accentuated (fig. 24) and artifacts will be introduced due to high frequency ringing or aliasing (see Hunt, 1978 for an extensive discussion of image deblurring). In some cases, where no high frequency is present, a small amount of random noise introduced into the high frequency domain will, on transformation, appear to sharpen an image.

An additional use of the 2D FFT that might not be immediately obvious is enlarging images. The method is remarkably simple. The transform of an image is formed and enlarged to the desired expansion size by filling with zeroes at the high frequencies. The process can be visualized as follows. An augmented (enlarged) array of zero values is formed whose size matches the desired output image. The four quadrants of the transform of the original image (see the left hand side of figure 1) are then placed (added to) the four respective corners of the enlarged array. This results in zeroes being inserted at the high frequencies of the transform. The augmented image is then produced by inverting the transform. An example is shown (fig. 25) for an enlargement of part of a TM scene of the Canon City, Colorado area using enlargement factors of 5 and 25. Results using a pixel replication and a standard bilinear algorithm are shown for comparison. At a factor 5, enlargement the bilinear algorithm looks fuzzy compared with the replication and the FFT methods. The disadvantages of the replication algorithm are more evident at a factor 25 enlargement. The FFT method yields the most accurate enlargement possible because, unlike the other two methods, it is designed not to distort the scene high frequency content. In addition to uniform enlargement, the FFT method can also be used to enlarge differentially in the two orthogonal directions. One obvious limitation for large images is the required memory and processing time to form the transform and its inverse.

The current attention to Geographic Information Systems (GIS) has focussed greater interest on the use of co-registered images from different data sources. Manual processing is often tedious, however, requiring visual comparison between images and selection of many control points. If the images are quite different (e.g. thermal day and night or radar and reflectance), then automatic techniques are not directly feasible. If the images look similar (e.g. reflectance data from Landsat TM, SPOT and AVIRIS) or can be processed to look similar (radar and digital terrain), then cross-correlation can be used to determine the offsets. The 2D FFT provides a fast means to compute the cross-correlation.

The general approach can be outlined as follows. Because multiplication in the frequency domain is equivalent to convolution in the spatial domain, and because correlation and convolution are closely related mathematically, the FFT can be used to compute discrete cross-correlation of images. Two augmented images are formed: one containing the first image in its upper left hand corner and the other containing the second image in its lower right hand corner. Each augmented array is chosen large enough to contain both input images in the opposite corners without overlap. The augmented arrays are then transformed, and a product formed of the transform of the first array and the complex conjugate of the second transformed array. Lastly, the correlation is computed from the inverse transform of the product array. The relationship is most easily examined by considering the one-dimensional case (Brigham, p 206, 1974) and extending that result to two dimensions.

The position of the maximum element in the correlation array provides the offset information and its value is a measure of the goodness of the correlation (and can be used as a weight in the registration). The algorithm can be applied to co-register two images by computing multiple correlations of smaller image segments and using these to determine a grid of control points.

Two satellite data sets (Landsat TM and SPOT) of an area near Canon City, Colorado are used to demonstrate the method (fig. 26). Because the SPOT data have a nominal resolution of 10m compared to the 30m resolution of the TM data, the SPOT data were first resampled every 3 pixels and lines. Fastest processing of the FFT algorithm occurs for image sizes that are powers of 2. For this example, 64 pixel by 64 line image segments were used. The resampled SPOT image (fig. 26A) was used as the control image. The TM image (fig. 26B) was visually windowed to roughly correspond to the SPOT image. Black fiducial marks have been applied to all images to assist in the comparison (normally this is done interactively by switching images back and forth on the video screen).

The control points computed by the FFT correlation were determined on a regular grid and fitted to an affine transformation. This transformation was applied to produce a registered TM image (fig 26D) using a nearest neighbor algorithm. Although the solar illumination between the two data sets are not the same (they were acquired about 13 months apart), the automatic registration appears to be fairly satisfactory. The white fiducial marks most clearly illustrate the effectiveness of the registration.

General implementation of this algorithm for image registration will be most efficient if the offsets (and hence the size of the correlation windows) are small. This can be achieved by first using an affine transformation to remove rotations and then sampling the images and applying the method iteratively.

Another use of the 2D FFT is to separate aspects of the image that occupy different spatial frequency domains. Structural patterns that are expressed topographically, such as a dominant directional grain, have a distinct domain. Topography is often associated with higher spatial frequencies than albedo variations, suggesting that the transform might be useful for separating these properties. To illustrate this separation a simple filter was applied to a SPOT image of the Canon City area to separate the low and high frequency components (fig. 27). A narrow sinc function was used in the decomposition to reduce the possibility of artifacts and the high frequency component (fig. 27C) was offset from zero by using the same dc term as in the original (fig. 27A) and the low frequency (fig. 27B) images. The decomposition is complete in that a sum of the low and high frequency images (the latter minus the dc term) yields the original image. As noted previously the low frequency (albedo component) looks fuzzy due to the lack of high frequencies.

Because discrete images can be represented by matrices, then vector and matrix algebra methods can be used directly to process the images (see Pratt, 1978, p121-139). The 2D FFT is routinely used in potential field studies for such operations as reduction-to-the-pole, vertical derivatives, upward and downward continuation, pseudo-gravity transformation, and trend analysis (eg. Hildenbrand, 1983). Potential field data often have different characteristics than most aircraft and space images, being sampled on irregularly spaced grids and often displaying much longer correlation lengths. Application of the techniques discussed here to these data involve examination of issues not discussed in this paper (see Cordell and Grauch, 1982). Lastly, the 2D FFT can also be used for fractal analysis and to estimate the power spectrum of an image for roughness characterization.

## Summary

The 2D FFT method is a powerful image processing tool, which can now be implemented at all levels ranging from PCs and workstations to mainframes. The transform of an image provides an effective means to identify regular spatial patterns and its primary use is to remove periodic noise from digital images. Although the mathematical form of the transform is simple, many relationships are most easily illustrated by example. Using these results a general set of filters was developed and tested using a variety of noisy satellite and aircraft images. Based on these and other examples, a general strategy is proposed for filtering images: filter derived images (not the original images from which they are formed), remove obvious noise in the transform, and apply the "minimum filter" necessary.

The 2D FFT algorithm can also be used as a general purpose spatial analysis tool for images. Examples show its use to enlarge images (particularly for large enhancement factors), to perform automatic geometric registration of images, and to extract albedo and slope information. Other uses include instrument response correction, spatial pattern analysis (feature extraction, structural domains, etc.). These uses are particular to images.

## Bibliography:

Bracewell, R., 1986, The Fourier transform and its applications: Second Edition, Revised, McGraw-Hill, New York, 474 p.

Brigham, E. O., 1974, The Fast Fourier Transform: Prentice-Hall, Inc., New Jersey, 252 p.

Clement, W. G., 1973, Basic Principles of two dimensional digital filtering, Geophysical Prospecting v 21, p 125-145.

Cordell, L., and Grauch, V. J. S., 1982, Reconciliation of the discrete and integral transform Fourier transforms: Geophysics, v 47, p 237-243.

Deregowski, S. M., 1971, Optimum digital filtering and inverse filtering in the frequency domain: Geophysical Prospecting v 19, p 729-768.

Eliason, P. T., Soderblom, L. A., and Chavez, P. S. Jr., 1981, Extraction of topographic and spectral albedo information from multispectral images, Phot. Engin. and Remote Sensing, v 48, n 11, p 1571-1579.

Gillespie, A. R., 1980, Digital techniques of image enhancement: Chap. 6 in Remote sensing in Geology, (Siegal, B. S. and Gillespie, A. R., eds.): John Wiley & Sons, New York, 702 p.

Hildenbrand, T. G., 1983, FFTFIL: A filtering program based on two-dimensional Fourier analysis: U. S. Geol. Survey Open-File Report 83-237, 30 p.

Hunt, B. R., 1978, Digital image processing: Chap. 4 in Applications of digital signal processing, (Oppenheim, A. V., ed): Prentice-Hall, New Jersey, 499 p.

Livo, K. E., 1990, REMAPP - PC, Remote sensing image processing software for MS-DOS Personal Computers Version 1.00: U.S. Geological Survey Open-File Report 90-88A-E, 58 p.

Oppenheim, A. V., and Schafer, R. W., 1975, Digital signal processing: Prentice-Hall, Inc, New Jersey, 585 p.

Pratt, W. K., 1978, Digital image processing: John Wiley & Sons, New York, p 235.

Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T., 1986, Numerical Recipes : the art of scientific computing: Cambridge University Press, Cambridge, England, p 451-453.

Rose, J. F., 1989, Spatial interference in the AVIRIS imaging spectrometer: Phot. Engin. and Remote Sensing, v 55, n 9, p 1339-1346.

Sawatzky, D. L., 1985, Programmer's Guide to REMAPP, REMote sensing Array Processing Procedures: U.S. Geological Survey Open-File Report 85-231, 21 p.

Singleton, R. C., 1967, A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage: IEEE Trans., Audio and Electroacoustics: v. AU-15, p 91-98.
Singleton, R. C., 1969, An algorithm for computing the mixed radix fast Fourier transform: IEEE Trans., Audio and Electroacoustics: v. AU-17, p93-103

**APPENDIX A. Fft2d.filter program**

```
c      input image(s)-> compute 2D FFT(s)
c      apply filters  [optional output of filtered transform]
c      transform back -> output image(s)
c
c      The Cooley-Tukey fast fourier transform in usasi basic fortran
c      is used.  The roots of the algorithm are somewhat ambiguous.  It probably begins with an Algol program
c      (Singleton R. C., 1967, IEEE Audio Trans. v AU-15, p 91-98) and was later modified by N. Brenner of c
c      Lincoln Labs and incorporated into the NCAR Library Routines (Adams, J. C. and Rotar, P., 1971, NCAR-c
c      TN/IA-67) 1967).  Filter currently available are
c      lowpass, highpass, harmonic, angular wedge, rectangle, strip, point and btub.
c      The btub filter is called U-shaped in the text.This program uses Remapp routines and conventions.
c      These are  described in a programmer's guide (Sawatzky, 1985) and
c      there is also a PC version (Livo, 1989).
c      The coordinate system for the 2DFFT is different from that of the
c      optical transform (e.g. see Hummer-Miller, 1990 Photo. Eng., v5, n1, p50).  The
c      The dc term(zero frequency) is in the upper left corner (not the center) and the
c      symmetry is a little less obvious (see comment 3 following).
c
c         ------> x                 -------> wx
c         |                         |
c         | spatial                 | fourier
c         | domain                  | domain
c         |                         |
c         v                         v
c         y                         wy
c
c            DC      --> wx              -wx <--
c             -----------------------------------
c             |                |                |
c           | |                |                |
c           | |                |                |
c           v |       I        |       II       |
c         -wy |                |                |
c             |                |                |
c             |                |                |
c             |                |                |
c             ------------------------------------ wNy (Nyquist)
c             |                |                |
c             |                |                |
c             |                |                |
c             |                |                |
c          wy |       IV       |       III      |
c             |                |                |
c           ^ |                |                |
c           | |                |                |
c           | |                |                |
c             -----------------------------------
c                              |
c                             wNx  (Nyquist)
c                     2D FFT Transform
c
```

```
c             ----------------wNy---------------
c             |                |                |
c             |                ^                |
c             |               wy                |
c             |       III             IV        |
c             |                |                |
c             |                |                |
c             |                |                |
c             |                |              wNx
c             ---------------DC------ wx >------
c             |                |                |
c             |                |                |
c             |                |                |
c             |                |                |
c             |       II       |       I        |
c             |                |                |
c             |                |                |
c             |                |                |
c             |                |                |
c             ----------------------------------
c
c                      Optical Transform
c
c .........................................................
c
c      Current input image settings are:
c       i)  up to 1024 pixel/scan lines and number of lines
c           such that lines*pixels <= 736*1200
c       ii) up to 6 input images with same set of filter(s)
c
c       N.B. To change image size you will need to modify:
c         1. all COMMON statements specifying data(both main program,
c                and appropriate subroutines)
c         2. the fp & fl arrays (main and subroutines)
c         3. the size test performed after reading in images.
c
c      The form of filter input is
c
c           n                    (number of filters to be applied)
c           opt1,i1,j1,k1,l1  (add 0's as necessary to supply 5 values)
c           opt2,i2,j2,k2,l2
c           ................
c           optn,in,jn,kn,ln  (il,j1 .... ln are filter perameters)
c        Program currently handles up to 50 filters at one time. (Easily
c          modified ... see iopt,iarg in integer statement.)
c
c      Comments:
c
c   1.  When more than one image is input, this program
c         forms pairs in the complex array "data" for faster processing
c          - one in the real & the other in the imaginary pert
c         of data.  Do NOT use peirs when computing just the
c         transforms ie. enter as single image cases.
c
c   2.  The fortran 1D format is used i.e.
c               2D                      1D
c         data(pixel,line) = data((line-1)*npix+pixel).  where
c                  npix - number of pixels/line.
c
```

```
c   3.  Transform symmetry:
c          For pixel p and line l of an mpix by nlin transform
c          data(kk)=data(k)
c
c          where k=(l-1)*npix+p
c          and   kk=(l'-1)*npix+p'
c          with p'=npix+2-p if p>1
c                 =1        if p=1
c               l'=mlin+2-l if l>1
c                 =1        if l=1
c
c   4.  Filter types:
c         - 3 general classes of filters:
c                   patch .....  interpolate across boundaries
c                 blocking ....  set interior & edges to zeroes
c              multiplicative..  * sinc function to smooth
c
c         - 9 types of filters: rectangle, btub, strip, point, angular
c             wedge, lowpass, highpass, harmonic and periodic boxes.
c
c   5.  A patch is constructed by interpolation of the transform
c         from its boundary values.  It is useful for reducing a
c         noise spike but, unlike a blocking filter, does not
c         create zeroes in the transform.  The interpolation is
c         bilinear with weights based on proximity.
c
c   6.  Blocking sets all spacified area including the boundary
c         to zero.  (Often this is the most effective way to
c         remove noise.)
c
c   7.  Multiplicative uses a sinc function to generate a
c         smooth roll off at the edges.  (This is the best way
c         to avoid filter artifacts.)
c
c   8.  Filter parameters:(4 values;fill with zeroes as necessary)
c
c      a. rectangle ... requires p1,p2;l1,l2 the starting &
c                              ending pixels and lines.
c      b.  strip ... requires p1,p2 or l1,l2 and applies to an entire
c                      line either vertical or horizontal.
c      c.  point ... requires p,l
c      d.  btub ... requires p,l,itub.
c             There are 2 orientation options:along line(itub=1) or
c                  pixel axis(itub=0).
c           This filter cuts off high frequencies on either the
c              line frequency or pixel frequency axis and is shaped
c              like a bathtub cut across its short dimension.  The
c              long axis of the bathtub is oriented along either
c              coordinate axis, and p,l specify the closest corner of the
c              bathtub.
c      e.  harmonic ... requires w,theta,nharm,dl
c               to remove a periodic set of vertical noise spikes
c               that appear en-echelon across the transform.
c         where
c           w is the periodic frequency of the pattern in pixel units;
c           theta - the angle(deg) made by the en-echelon pattern;
c                 same measuring convention as for wedge
c           nharm is the max number of harmonics;
c           and dl is the half length (in pixel units) of the noise
c             pattern in the line direction.
c      integer scaling: w and theta scaled by factor 100
c               If dl is assigned a negative value then the entire line
c               is blocked.
```

```
c      f.  low and high pass - requires rho1 the cutoff in pixel units.
c           The high pass allows for an optional enhancement factor,
c           entered as a %. A default value of 50% is used if this
c           this parameter is entered as 0. To set it to 75% enter 75.
c
c      g.  angular wedge ... requires theta,delt,rho1,rho2
c             and blocks out a wedge in polar coord space.
c             theta is measured clockwise(deg) from the line(vertical)
c             axis.  If the origin of the wedge in transform
c             space is in the upper right corner [or by symmetry
c             the lower left} the angle is +ve.
c             The angular width of the wedge is delt(deg) and it is
c             radially bounded between rho1 and rho2..
c     integer scaling: factor 10 for all 4 parameters.
c
c      h.  periodic boxes ... along line axis
c             ... requires lstart,dl,halfwidth,number of boxes

c    9.              Filter options table:
c
c     filter         patch       block        mult      special
c    rectangular       1           6           7
c    strip[line]       3           9          10
c    strip[pixel]     15          14           -
c    point             4           8           -
c     btub             2          11           5      itub=1 (line axis)
c                                                          =0 (pixel axis)
c     wedge            -           -          16
c    harmonic         17          13          18
c    low pass          -           -          12
c    high pass         -           -          19
c    periodic box     20          21           -
c
c  *  output log(transform) of modulus         -1  ..only use with i2=0
c  *  output transform                         -2
c      skip filtering                           0  {to test i/o}
c
c  *  use i2=0 modulus(amplitude)
c           =1 real
c           =2 imaginary
c           =3 phase
c       ex. -2,1,0,0,0    will output the real part of the transform.
c           -1,0,0,0,0         "      the log(modulus)    "
c  N.B. The filter option (-1 or -2) to output the transform MUST
c         be the LAST filter option in the list in order to include
c         the result of all filters in the output.
c   10. When adding new filters to this program do NOT assume the
c         transform symmetry (see #3).  Apply filters to symmetrical
c         pairs using symmetry function. (see subroutine symm)
c...........................................
c
```

```
      dimension nn(2),work(4000)
      complex data
      common data(883200)
      integer iopt(50),iarg(50,4)
      integer*2 array(1024),array1(1024)
      integer fcbi(256,6),fcbo(256,6),jarg(4)
      data ndim/2/,fcbi,fcbo/3072*0/,jarg/4*0/
c
c      ************* open i/o files *****************
c
      write(6,*)'current version as of 5/8/91'
      write(6,*)'enter number[<7] of input image files)'
      read(5,*)ifileno
      do 1 i=1,ifileno
      ik=i+6
      call diskio(0,ik,0,fcbi(1,i))
      if(fcbi(1,i).lt.0)call exit(i)
      npix=fcbi(2,1)
      mlin=fcbi(3,1)
c      .....       image size test     .......
       if(npix*mlin.gt.883200)then
      write(6,*)' incorrect input size: npix*mlin=',npix*mlin
      write(6,*)' max allowed size is 736*1200=883200'
      stop
      end if
c      .....                           .......
      nn(1)=npix
      nn(2)=mlin
      j=i+ifileno
      jk=j+6
      fcbo(1,i)=8
      fcbo(2,i)=npix
      call diskio(0,jk,0,fcbo(1,i))
 1     continue
c
c     ****************** enter filter parms *******
c
c      to check i/o without filtering let nfilter=1 and use 0,0,0,0,0.
c
      write(6,*)'enter number of filters'
      read(5,*)nfilter
c     write(6,*)'number of filters is:',nfilter
      write(6,*)'enter filter option,arg1,arg2,...'
      do 2 i=1,nfilter
      read(5,*)iopt(i),iarg(i,1),iarg(i,2),iarg(i,3),
     1 iarg(i,4)
 2    continue
      if(iopt(nfilter).lt.0.and.ifileno.gt.1)then
      write(6,*)'It is NOT permitted to compute and output'
      write(6,*)'multiple transforms. Enter one at a time.'
      write(6,*)'See comment #1 in program listing.'
      stop
      end if
c
c     ***************** read in images ************
c
      do 1000 ifile=1,ifileno,2
      if(ifileno.gt.1)then
      jj=ifile+1
      write(6,*)'reading in file pair: ',ifile,' and ',jj
      end if
      if(ifileno.eq.1)write(6,*)'reading in file'
      ifileo=ifile+ifileno
       ifilek=ifile+6
       do 7 j=1,mlin
       call diskio(9,ifilek,array,fcbi(1,ifile))
       call unpack(array,fcbi(1,ifile))
       jfile=ifile+1
```

```
        jfilek=jfile+6
        jfileo=jfile+ifileno
        if(ifile.lt.ifileno)then
        call diskio(9,jfilek,array1,fcbi(1,jfile))
        call unpack(array1,fcbi(1,jfile))
        end if
         do 10 i=1,npix
         k=(j-1)*npix+i
         if(ifile.lt.ifileno)then
         data(k)=cmplx(array(i),array1(i))
         else
         data(k)=cmplx(array(i),0.)
         end if
 10      continue
 7      continue
c
c       *************** forward transform **************
c
       write(6,*)'forming transform'
       call fourt(data,nn,ndim,-1,1,work)
c
c       *************** apply filters ******************
c
       write(6,*)'applying ',nfilter,' filter(s).'
         iswitch=0
       do 11 ifilter=1,nfilter
         ifopt=iopt(ifilter)
         ip1=iarg(ifilter,1)
         ip2=iarg(ifilter,2)
         il1=iarg(ifilter,3)
         il2=iarg(ifilter,4)
        if(ifopt.le.-1)iswitch=1
c
        if (ifopt.eq.20.or.ifopt.eq.21.or.ifopt.eq.22)then
         lstart=ip1
         idl=ip2
         ihalfwidth=il1
         nbox=il2
         call periodbox(lstart,idl,ihalfwidth,nbox,ifopt,npix,mlin)
        end if
c
        if (ifopt.eq.16)then
         theta=ip1*.1
         dthet=ip2*.1
         rho1=il1*.1
         rho2=il2*.1
         call wedge(theta,dthet,rho1,rho2,npix,mlin)
        end if
c
        if(ifopt.eq.3.or.ifopt.eq.9.or.ifopt.eq.10.or.
     1   ifopt.eq.14.or.ifopt.eq.15)then
         call strip(ip1,ip2,il1,il2,ifopt,npix,mlin)
        end if
c
        if (ifopt.eq.13.or.ifopt.eq.17.or.ifopt.eq.18)then
         w=ip1*.01
         t=ip2*.01
         nhmax=il1
         ndl=il2
         call harmonic(w,t,nhmax,ndl,ifopt,npix,mlin)
        end if
c
        if(ifopt.eq.12)call lowpass(ip1,npix,mlin)
c
        if(ifopt.eq.19)then
        factor=ip2*.01
        if(factor.eq.0)factor=.5
        call highpass(ip1,factor,npix,mlin)
        end if
```

```
c
        if(ifopt.eq.1.or.ifopt.eq.6.or.ifopt.eq.7)call rectangle(
     1   ip1,ip2,il1,il2,ifopt,npix,mlin)
c
        if(ifopt.eq.2.or.ifopt.eq.5.or.ifopt.eq.11)call btub(
     1   ip1,ip2,il1,ifopt,npix,mlin)
c
        if(ifopt.eq.4.or.ifopt.eq.8)call point(ip1,ip2,ifopt,
     1   npix,mlin)
c
 11    continue
c
       if(iswitch.eq.1)then
        write(6,*)'forming re-scaled transform'
        zmin=1.E11
        zmax=-1.E11
        do 888 j=1,mlin
        do 888 i=1,npix
         k=(j-1)*npix+i
       if(i.eq.1.and.j.eq.1.and.real(data(1)).gt.real(mlin*npix))
     1  go to 888
        if(ip1.eq.0)z=sqrt(real(data(k))**2+aimag(data(k))**2)
        if(ip1.eq.1)z=real(data(k))
        if(ip1.eq.2)z=aimag(data(k))
        if(ip1.eq.3)z=atan2(real(data(k)),aimag(data(k)))
        if(ifopt.eq.-1)then
        if(z.lt.0.)then
        write(6,*)'You are trying to compute the log of a -ve #'
        write(6,*)'Either you are performing an illegal option'
        write(6,*)'or this program needs to be checked.  SEE '
        write(6,*)'Ken Watson for further details.'
        write(6,*)'This program has now terminated.'
        stop
        end if
          if(z.eq.0.)then
           z=-alog(2.)
          else
           z=alog(z)
          end if
        end if
c        when the dc term is too large ..... reset
         if((ip1.eq.0).and.(i.eq.1).and.(j.eq.1))goto 888
         zmin=amin1(zmin,z)
         zmax=amax1(zmax,z)
 888    continue
        if(zmin.eq.zmax)then
        write(6,*)'zmin.eq.zmax .... no stretch possible... paused.'
        pause
        end if
        a=255./(zmax-zmin)
        b=-a*zmin
        do 889 j=1,mlin
        do 890 i=1,npix
         k=(j-1)*npix+i
        if(ip1.eq.0)z=sqrt(real(data(k))**2+aimag(data(k))**2)
        if(ip1.eq.1)z=real(data(k))
        if(ip1.eq.2)z=aimag(data(k))
        if(ip1.eq.3)z=atan2(real(data(k)),aimag(data(k)))
         if(ifopt.eq.-1)then
          if(z.eq.0.)then
          z=-alog(2.)
          else
          z=alog(z)
          end if
         end if
         array1(i)=anint(a*z+b)
         if(array1(i).lt.0)array1(i)=0
         if(array1(i).gt.255)array1(i)=255
 890    continue
```

```
       call pack(array1,fcbo(1,ifile))
       ifileok=ifileo+6
       call diskio(10,ifileok,array1,fcbo(1,ifile))
  889  continue
       write(6,*)'closing files.'
       ifilek=ifile+6
       ifileok=ifileo+6
       call diskio(6,ifilek,0,fcbi(1,ifile))
       call diskio(6,ifileok,0,fcbo(1,ifile))
        if(ifile.lt.ifileno)then
         jfilek=jfile+6
         jfileok=jfileo+6
         call diskio(6,jfilek,0,fcbi(1,jfile))
         call diskio(6,jfileok,0,fcbo(1,jfile))
        end if
        go to 1000
      end if
c
        write(6,*)'forming inverse transform'
        call fourt(data,nn,ndim,+1,1,work)
c
        write(6,*)'output filtered images'
        do 300 j=1,mlin
         do 301  i=1,npix
         k=(j-1)*npix+i
         data(k)=data(k)/(nn(1)*nn(2))
         array1(i)=anint(aimag(data(k)))
         if(array1(i).gt.255)array1(i)=255
         if(array1(i).lt.0)array1(i)=0
         array(i)=anint(real(data(k)))
         if(array(i).gt.255)array(i)=255
         if(array(i).lt.0)array(i)=0
 301     continue
        if(ifile.lt.ifileno)then
         call pack(array1,fcbo(1,jfile))
         jfileok=jfileo+6
         call diskio(10,jfileok,array1,fcbo(1,jfile))
        end if
        call pack(array,fcbo(1,ifile))
        ifileok=ifileo+6
  300   call diskio(10,ifileok,array,fcbo(1,ifile))
c
c     ***************  close files  *******************
c
        write(6,*)'closing files.'
        ifilek=ifile+6
        ifileok=ifileo+6
        call diskio(6,ifilek,0,fcbi(1,ifile))
        call diskio(6,ifileok,0,fcbo(1,ifile))
        if(ifile.lt.ifileno)then
        jfilek=jfile+6
        jfileok=jfileo+6
        call diskio(6,jfilek,0,fcbi(1,jfile))
        call diskio(6,jfileok,0,fcbo(1,jfile))
        end if
 1000 continue
      write(6,*)'finished.'
      end
```

```
c
c       ***********************************************
c
c                        SUBROUTINES
c
c       ***********************************************
c
        subroutine halfsinc(i1,i2,f)
c        this is a rapid rolloff filter between i1 and i1+3
        dimension f(1200)
        data beta/.821534976379/,gamma/4.49340945791/
c       note f(x)=1-beta+beta*sin(x*gamma)/(x*gamma)
c            f(0)=1
c            f(1)=0
c           df/dx = 0  at x=1
c
c           x=(i-i1)/(i2-i1)   if i2<i1+4
c            =(i-i1)/4         if i2>i1+3
c                   i=i1,i2
c
         do 8000 k=1,2000
 8000     f(k)=0.
        f(i2)=0.
        do 2 i=1,i1
  2     f(i)=1
        do 1 i=i1+1,i1+3
        if(i2.gt.il+3)then
        x=float(i-i1)/4.
        else
        x=float(i-i1)/float(i2-i1)
        end if
        f(i)=0.
        if(x.lt.1.)f(i)=1.-beta+beta*sin(gamma*x)/(gamma*x)
  1     continue
        return
        end
c
        subroutine smooth(i1,npix,f)
c        this is a gradual rolloff filter
        dimension f(1200)
        data beta/.821534976379/,gamma/4.49340945791/
        i2=npix/2+1
        f(i1)=1.
        f(i2)=0.
        do 1 i=i1+1,i2-1
        x=float(i-i1)/float(i2-i1)
        f(i)=1.-beta+beta*sin(gamma*x)/(gamma*x)
  1     continue
        return
        end
c
        subroutine symm(ipix,ilin,npix,mlin,kout)
c        to compute symmetry point and return value in 1D parameter kout.
        ip1=npix+2-ipix
        if(ipix.eq.1)ip1=1
        il1=mlin+2-ilin
        if(ilin.eq.1)il1=1
        kout=(il1-1)*npix+ip1
        return
        end
```

```
c
c
      subroutine periodbox(lstart,idl,ihalfwidth,nbox,ifopt,npix,mlin)
c       to filter periodic boxes along line axis
c       N.B. nbox refers to number in first quadrant only
c         symmetry will take care of the rest
c
c  options: = 20 patch; =21 blocking
       complex data
       common data(883200)
       write(6,*)'entering periodbox.'
       do 100 kbox=1,nbox
       lo=lstart+(kbox-1)*idl
       if(ifopt.eq.21.)then
       do 101 ip=-ihalfwidth+1,ihalfwidth+1
       ipix=ip
       if(ip.lt.1)ipix=npix+ip
       do 102 il=lo-ihalfwidth+1,lo+ihalfwidth+1
       ilin=il
       if(il.lt.1)ilin=mlin+il
         k=(ilin-1)*npix+ipix
         call symm(ipix,ilin,npix,mlin,ksymm)
         data(k)=0.
         data(ksymm)=0.
  102    continue
  101    continue
         end if
  100    continue
       return
       end
c
       subroutine rectangle(ip1,ip2,il1,il2,ifopt,npix,mlin)
c       to filter rectangular areas
c  options: = 1 patch, =6  blocking, =7 mult
       complex data
       common data(883200)
       dimension fp(1200),fl(1200)
         dp=ip2-ip1
        if(ifopt.ne.6)then
         if(il1.ge.(il2-1).or.ip1.ge.(ip2-1))then
          write(6,*)'bounding error at rectangular interp'
          write(6,*)'il1,il2:',il1,il2
          write(6,*)'ip1,ip2:',ip1,ip2
          stop
         end if
        end if
       if (ifopt.eq.7)then
        imidl=int(float(il2+il1)/2.)
        imidp=int(float(ip2+ip1)/2.)
        call halfsinc(ip1,imidp,fp)
         do 666 i=imidp+1,ip2
          j=ip1+ip2-i
 666     fp(i)=fp(j)
         call halfsinc(il1,imidl,fl)
         do 667 i=imidl+1,il2
         j=il1+il2-i
 667     fl(i)=fl(j)
       end if
c           ..... routine  .....
         do 12 ilin=il1,il2
         do 12 ipix=ip1,ip2
         if (ifopt.ne.6)then
         c1=float(ipix-ip1)/dp
         c2=float(ip2-ipix)/dp
         end if
         k=(ilin-1)*npix+ipix
         k1=(ilin-1)*npix+ip2
         k2=(ilin-1)*npix+ip1
         call symm(ipix,ilin,npix,mlin,kout)
```

```
          call symm(ip2,ilin,npix,mlin,k1out)
          call symm(ip1,ilin,npix,mlin,k2out)
          if(ifopt.eq.1)then
          data(k)=c1*data(k1)+c2*data(k2)
          data(kout)=c1*data(k1out)+c2*data(k2out)
          end if
          if(ifopt.eq.6)then
          data(k)=0.
          data(kout)=0.
          end if
       if(ifopt.eq.7)then
       factor=fp(ipix)+fl(ilin)-fp(ipix)*fl(ilin)
       data(k)=factor*data(k)
       data(kout)=factor*data(kout)
       end if
 12    continue
        return
        end
c
        subroutine wedge(theta,dthet,rho1,rho2,npix,mlin)
c        to filter wedge shaped areas ie bounded in theta & r
        complex data
        common data(883200)
         iswitch=0
         if (theta.lt.0.) iswitch=1
         theta=abs(theta)
         ip1=int(rho1*sind(theta-dthet*.5))
         ip2=int(rho2*sind(theta+dthet*.5))+1
         if (iswitch.eq.1)then
         ip1a=npix+1-ip1
         ip1=npix+1-ip2
         ip2=ip1a
         end if
         il1=int(rho1*cosd(theta+dthet*.5))
         il2=int(rho2*cosd(theta-dthet*.5))+1
         do 1849 ip=ip1,ip2
         do 1849 il=il1,il2
         if(iswitch.eq.0)rho=sqrt((ip-1.)**2+(il-1.)**2)
         if(iswitch.eq.1)rho=sqrt((npix-ip)**2+(il-1.)**2)
         if((rho.lt.rho1).or.(rho.gt.rho2))goto 1849
         if(iswitch.eq.0)phi=atand((ip-1.)/(il-1.))
         if(iswitch.eq.1)phi=atand((npix-ip)/(il-1.))
         if((phi.lt.theta-dthet*.5).or.(phi.gt.theta+dthet*.5))
      1 goto 1849
         data((il-1)*npix+ip)=0.
         call symm(ip,il,npix,mlin,ksymm)
         data(ksymm)=0.
1849        continue
        return
        end
c
        subroutine harmonic(w,t,nhmax,ndl,ifopt,npix,mlin)
c           periodic angular
c       enter w,t,nhmax,ndl
c       where w is the angular period in pixel units
c             t is the angle in degrees
c                +ve if pattern starts at origin
c                -ve if it starts at p=1,l=mlin
c             nhmax is the max no of harmonics to filter
c             ndl the half line length of the petch filter
c        ndl<0 employ a full line block at harmonic
        complex data
        common data(883200),fp(1200),fl(1200)
        integer pn
         iline=nhmax
         nhmax=abs(nhmax)
         ntest=ndl
         ndl=abs(ndl)
         sint=sind(abs(t))
        cost=cosd(t)
        do 1999 nh=1,nhmax
         ln=1+mod(int(w*nh*cost+.5),mlin)
         if(t.ge.0.)then
        pn=1+mod(int(w*nh*sint+.5),npix)
         else
         pn=npix-mod(int(w*nh*sint+.5),npix)
         end if
        if(ifopt.eq.18)then
        call halfsinc(pn-4,pn,fp)
        do 1997 i=1,4
 1997   fp(pn+i)=fp(pn-i)
        call halfsinc(ln-50,ln,fl)
        do 1996 j=-ndl,ndl
        il=ln+j
 1996   fl(ln+j)=fl(ln-j)
        end if
        do 1998 il=ln-ndl,ln+ndl
        if(il.lt.1)il=mlin+il
        if(il.gt.mlin)il=il-mlin
        k=(il-1)*npix+pn
        call symm(pn,il,npix,mlin,ksymm)
       if(ifopt.eq.17)then
c             apply patch
        data(k-1)=(2*data(k-2)+data(k+2))/3.
        data(ksymm-1)=(2*data(ksymm-2)+data(ksymm+2))/3.
        data(k+1)=(data(k-2)+2*data(k+2))/3.
        data(ksymm+1)=(data(ksymm-2)+2*data(ksymm+2))/3.
        data(k)=(data(k-2)+data(k+2))/2.
        data(ksymm)=(data(ksymm-2)+data(ksymm+2))/2.
       end if
       if(ifopt.eq.13)then
c             apply block
        data(k-1)=0.
        data(ksymm-1)=0.
        data(k+1)=0.
        data(ksymm+1)=0.
        data(k)=0.
        data(ksymm)=0.
       end if
       if(ifopt.eq.18)then
c               apply mult
        kp=(mlin+1)*npix+2-k
        factor=fp(ip)+fl(il)-fp(ip)*fl(il)
        data(k)=factor*data(k)
c       write(6,*)'nh,p,l,factor:',nh,ip,il,factor
        data(kp)=factor*data(kp)
       end if
1998    continue
        if(ntest.lt.0)then
c               full line block
       idl=int(mlin/2)
c       2/3 line block
       if(iline.lt.0)idl=int(2*idl/3)
        do 3345 il=ln-idl,ln+idl
        if(il.lt.1)il=mlin+il
        if(il.gt.mlin)il=il-mlin
        k=(il-1)*npix+pn
        call symm(pn,il,npix,mlin,ksymm)
        data(k)=0.
        data(ksymm)=0.
 3345   continue
        end if
 1999   continue
        return
        end
```

```
c
        subroutine point(ip,il,ifopt,npix,mlin)
c    option = 4,8     patch,block
c
       complex data,sum
       common data(883200)
        kk=(il-1)*npix+ip
        call symm(ip,il,npix,mlin,kkout)
        sum=cmplx(0.,0.)
        sumout=cmplx(0.,0.)
        count=0
        do 3000 ipix=max0(ip-1,1),ip+1
        do 3000 ilin=max0(il-1,1),il+1
        if(ipix.eq.ip.and.ilin.eq.il)go to 3000
        count=count+1
        k=(ilin-1)*npix+ipix
        call symm(ipix,ilin,npix,mlin,kout)
        sum=sum+data(k)
        sumout=sumout+data(kout)
 3000   continue
         if(ifopt.eq.4)then
        data(kk)=sum/count
        data(kkout)=sumout/count
         else
         data(kk)=0.
         data(kkout)=0.
         end if
        return
        end
c
        subroutine lowpass(ip1,npix,mlin)
       complex data
       common data(883200)
       dimension fp(1200)
           call halfsinc(ip1,ip1+5,fp)
c          call smooth(ip1,npix,fp)
           ii1=int(npix/2)+1
           jj1=int(mlin/2)+1
           do 1444 i=1,ii1
           do 1444 j=1,jj1
        r=(sqrt((float(i)/npix)**2+(float(j)/mlin)**2))*npix
         if(int(r).lt.ip1)go to 1444
         k=(j-1)*npix+i
         i1=npix+2-i
         if(i.eq.1)i1=1
         j1=mlin+2-j
         if(j.eq.1)j1=1
         k1=(j-1)*npix+i1
         k2=(j1-1)*npix+i1
         k3=(j1-1)*npix+i
         if(int(r).lt.ip1+5)then
         factor=fp(int(r))
         data(k)=data(k)*factor
         data(k1)=data(k1)*factor
         data(k2)=data(k2)*factor
         data(k3)=data(k3)*factor
         else
         data(k)=0.
         data(k1)=0.
         data(k2)=0.
         data(k3)=0.
         end if
1444     continue
         return
         end
        subroutine highpass(ip1,enh_factor,npix,mlin)
       complex data
       common data(883200)
       dimension fp(1200)
           call halfsinc(ip1,ip1+5,fp)
           ii1=int(npix/2)+1
           jj1=int(mlin/2)+1
           do 2444 i=1,ii1
           do 2444 j=1,jj1
        r=(sqrt((float(i-1)/npix)**2+(float(j-1)/mlin)**2))*npix
         if(int(r).lt.ip1)go to 2444
         k=(j-1)*npix+i
         i1=npix+2-i
         if(i.eq.1)i1=1
         j1=mlin+2-j
         if(j.eq.1)j1=1
         k1=(j-1)*npix+i1
         k2=(j1-1)*npix+i1
         k3=(j1-1)*npix+i
         factor=1+enh_factor*(1-fp(int(r)))
         data(k)=data(k)*factor
         data(k1)=data(k1)*factor
         data(k2)=data(k2)*factor
         data(k3)=data(k3)*factor
2444     continue
         return
         end
c
        subroutine btub(ip1,il1,itub,ifopt,npix,mlin)
c    option = 2 patch, 5 mult , 11 block
c       itub=1 line axis;  =0 pixel axis
       complex data
       common data(883200)
       dimension fp(1200),fl(1200)
         il2=int(mlin/2.+1)
         ip2=int(npix/2.+1)
         if(ifopt.eq.5)then
c              mult filter     line axis
          if(itub.eq.1)then
          ixx=ip1-4
          if(ixx.lt.0)ixx=1
          call halfsinc(ixx,ip1,fp)
          do 668 i=1,ip1
 668      fp(i)=1-fp(i)
          call halfsinc(il1,il2,fl)
          else
c              mult filter      pixel axis
          call halfsinc(1,il1,fl)
          do 669 i=1,il1
 669      fl(i)=1-fl(i)
          call halfsinc(ip1,ip2,fp)
          end if
         end if
         if(itub.eq.1)then
c                              line axis
          do 13 ilin=il1+1,il2
         u1=ilin-il1
         u2=(mlin-il1+1)-ilin
         ilin1=mlin+2-ilin
         if(ilin.eq.1)ilin1=1
         do 13 ipix=1,ip1-1
         ipix1=npix+2-ipix
         if(ipix.eq.1)ipix1=1
         k=(ilin-1)*npix+ipix
         ka=(ilin-1)*npix+ipix1
         call symm(ipix,ilin,npix,mlin,kb)
         call symm(ipix1,ilin,npix,mlin,kc)
        if(ifopt.eq.5.or.ifopt.eq.11)then
        if(ifopt.eq.5)factor=fp(ipix)+fl(ilin)-fp(ipix)*
```

```
     1    fl(ilin)
        if(ifopt.eq.11)factor=0.
          data(k)=factor*data(k)
          data(ka)=factor*data(ka)
          data(kb)=factor*data(kb)
          data(kc)=factor*data(kc)
          go to 13
          end if
c            patch filter (still line axis)
         u3=ip1-ipix
         d=u2*u3+u1*u3+u1*u2
         w1=u2*u3/d
         w2=u1*u3/d
         w3=u1*u2/d
         k1=(il1-1)*npix+ipix
         k2=(mlin+2-il1-1)*npix+ipix
         k3=(ilin-1)*npix+ip1
         k3a=(ilin1-1)*npix+ip1
         call symm(ipix,il1,npix,mlin,k1out)
         call symm(ipix,mlin+2-il1,npix,mlin,k2out)
         call symm(ip1,ilin,npix,mlin,k3out)
         call symm(ip1,ilin1,npix,mlin,k3aout)
         data(k)=w1*data(k1)+w2*data(k2)+w3*data(k3)
         data(kout)=w1*data(k1out)+w2*data(k2out)+
     1 w3*data(k3out)
         data(ka)=w1*data(k2)+w2*data(k1)+w3*data(k3a)
         data(kaout)=w1*data(k2out)+w2*data(k1out)+w3*
     1 data(k3aout)
  13     continue
         else
c                      pixel axis
         do 33 ipix=ip1+1,ip2
         u1=ipix-ip1
         u2=(npix-ip1+1)-ipix
         ipix1=npix+2-ipix
         if(ipix.eq.1)ipix1=1
         do 33 ilin=1,il1-1
         ilin1=mlin+2-ilin
         if(ilin.eq.1)ilin1=1
         k=(ilin-1)*npix+ipix
         ka=(ilin1-1)*npix+ipix
         kb=(ilin-1)*npix+ipix1
         kc=(ilin1-1)*npix+ipix1
         call symm(ipix,ilin,npix,mlin,kout)
         call symm(ipix,ilin1,npix,mlin,kaout)
         call symm(ipix1,ilin,npix,mlin,kbout)
         call symm(ipix1,ilin1,npix,mlin,kcout)
         if(ifopt.eq.5.or.ifopt.eq.11)then
        if(ifopt.eq.5)factor=fp(ipix)+fl(ilin)-fp(ipix)*f1(ilin)
        if(ifopt.eq.11)factor=0.
          data(k)=factor*data(k)
          data(kout)=factor*data(kout)
          data(ka)=factor*data(ka)
          data(kaout)=factor*data(kaout)
          data(kb)=factor*data(kb)
          data(kbout)=factor*data(kbout)
          data(kc)=factor*data(kc)
          data(kcout)=factor*data(kcout)
          go to 33
         end if
         u3=il1-ilin
         d=u2*u3+u1*u3+u1*u2
         w1=u2*u3/d
         w2=u1*u3/d
         w3=u1*u2/d
         k1=(ilin-1)*npix+ip1
         k2=(ilin-1)*npix+npix+2-ip1
         k3=(il1-1)*npix+ipix
         k3a=(mlin+2-il1-1)*npix+ipix
        call symm(ip1,ilin,npix,mlin,k1out)
        call symm(npix+2-ip1,ilin,npix,mlin,k2out)
        call symm(ipix,il1,npix,mlin,k3out)
        call symm(ipix,mlin+2-il1,npix,mlin,k3aout)
        data(k)=w1*data(k1)+w2*data(k2)+w3*data(k3)
        data(kout)=w1*data(k1out)+w2*data(k2out)+w
     1  3*data(k3out)
        data(ka)=w1*data(k2)+w2*data(k1)+w3*data(k3a)
        data(kaout)=w1*data(k2out)+w2*data(k1out)+w
     1  3*data(k3aout)
  33    continue
       end if
       return
       end
       subroutine strip(ip1,ip2,il1,il2,ifopt,npix,mlin)
      complex data
      common data(883200)
      dimension fp(1200)
c         horizontal strip [pixel]       option 14,15
       if(ifopt.eq.14.or.ifopt.eq.15)then
       l1=ip1
       l2=ip2
       do 1789 ip=1,npix
       do 1789 il=l1,l2
       k=(il-1)*npix+ip
       k1=k+(l1-il-1)*npix
       k2=k+(l2-il-1)*npix
       call symm(ip,il,npix,mlin,ksymm)
       call symm(ip,l1-1,npix,mlin,ks1)
       call symm(ip,l2+1,npix,mlin,ks2)
       z=(il-l1+1.)/(l2-l1+2.)
       if(ifopt.eq.14)data(k)=0.
       if(ifopt.eq.15)data(k)=data(k1)+(data(k2)-data(k1))*z
       if(ifopt.eq.15)data(ksymm)=data(ks1)+(data(ks2)
     1 -data(ks1))*z
       if(ifopt.eq.14)data(ksymm)=0.
1789   continue
       end if
c   option = 3,9,10     vertical strip:patch,block,mult
       if((ifopt.eq.3).or.(ifopt.eq.9)
     1.or.(ifopt.eq.10))then
       if((ip1.eq.ip2).and.ifopt.ne.9)then
       ip1=ip1-1
       ip2=ip2+1
       end if
        if(ip1.gt.ip2)then
        write(6,*)'strip interp error, ip1.gt.ip2'
        write(6,*)'ip1,ip2:',ip1,ip2
        stop
        end if
        dp=ip2-ip1
         if(ifopt.eq.10)then
          imidp=int(float(ip2+ip1)/2.)
          call halfsinc(ip1,imidp,fp)
          do 766 i=imidp+1,ip2
           j=ip1+ip2-i
 766       fp(i)=fp(j)
        end if
         do 114 ilin=1,mlin
         ilin1=mlin+2-ilin
         if(ilin.eq.1)ilin1=1
         k1=(ilin-1)*npix+ip1
         k2=(ilin-1)*npix+ip2
         call symm(ip1,ilin,npix,mlin,k1out)
         call symm(ip2,ilin,npix,mlin,k2out)
         do 114 ipix=ip1,ip2
         c1=0
         c2=0
         if(ifopt.ne.9)then
      c1=float(ip2-ipix)/dp
      c2=float(ipix-ip1)/dp
      end if
      k=(ilin-1)*npix+ipix
      call symm(ipix,ilin,npix,mlin,kout)
      if(ifopt.eq.3)then
      data(k)=c1*data(k1)+c2*data(k2)
      data(kout)=c1*data(k1out)+c2*data(k2out)
      end if
      if(ifopt.eq.10)then
      data(k)=fp(ipix)*data(k)
      data(kout)=fp(ipix)*data(kout)
      end if
      if(ifopt.eq.9)then
      data(k)=0
      data(kout)=0
      end if
114   continue
    end if
      return
      end
```

## APPENDIX B. 2D FFT of some fundamental patterns

**Image impulse.**

Define an image impulse (see figure 2A) at pixel *j'* (measured horizontally from the left edge) and at line *k'* (measured vertically from the top edge) :

$$\Delta(j',k')=\Delta(j-j').\Delta(k-k') \quad \text{where } \Delta(p-p')\equiv 1 \;\; p=p' \quad \equiv 0 \text{ otherwise.} \tag{3}$$

Then its transform in u,v space (where u is parallel to j and $\nu$ is parallel to k) is given by

$$\Delta_T=\frac{1}{N}e^{-\frac{2\pi i}{N}(uj'+vk')}, \tag{4}$$

and the real part of the transform is

$$\Re(\Delta_T)=\frac{1}{N}\cos(\frac{2\pi}{N}(uj'+vk')). \tag{5}$$

This is a sinusoidal pattern (see figure 2B) of dark and light bands. The distance between adjacent bands, and the slope of the line passing though one of the bands is easily determined from the parametric equation

$$\frac{2\pi}{N}(uj'+vk') = constant + 2\pi l \qquad l = 0, 1, 2... \quad \text{where } D = \frac{N}{\sqrt{(k'^2+j'^2)}} \text{ and } \theta = \tan^{-1}(-k'/j'). \tag{6}$$

**Periodic pattern of impulses along edge.**

Consider an equally spaced set of impulse functions, separated by a distance $\sigma$, along the left hand edge of an image (see figure 7C). Then

$$I(\sigma) = \Delta(j).\Delta(k-s\sigma) \qquad s=0, 1, 2.., N/\sigma-1 \tag{7}$$

and its transform is

$$\therefore I_T = \frac{1}{N}\sum_{s=0}^{N/\sigma-1} e^{\frac{-2\pi i v s\sigma}{N}} = \begin{cases} 1/\sigma & \text{if } \nu = \text{integer} \times N/\sigma \\ 0 & \text{if } \nu \neq \text{integer} \times N/\sigma. \end{cases} \tag{8}$$

This is a series of parallel lines (see figure 7D) with spacing N/$\sigma$.

**Image "line".**

If we define an image "line" as a contiguous array of image impulses across the image (see figure 9A) then a line at k=k' is given by

$$L(k') = \Delta(k-k').\Delta(j-s) \text{ where } s = 0,\ 1,\ 2...,\ N-1 \tag{9}$$

and the transform is

$$\begin{aligned} L_T &= \frac{1}{N}\sum_{s=0}^{N-1} e^{\frac{-2\pi i}{N}(us+vk')} \\ &= e^{-\frac{2\pi i v k'}{N}}\Delta(u). \end{aligned} \tag{10}$$

The real part is given by,

$$\Re(L_T) = \cos(\frac{2\pi v k'}{N})\Delta(u), \tag{11}$$

which is a sinusoidal pattern along the vertical axis (see figure 9B).

**Periodic line pattern.**

For a periodic line pattern (see figure 9C)

$$P(\sigma) = \Delta(k-r\sigma).\Delta(j-s) \text{ where } r = 0,\ 1...,\ \bar{r};\ s = 0,\ 1,..,\ N\text{-}1. \tag{12}$$

The transform is

$$\begin{aligned} P_T &= \frac{1}{N}\sum_{r=0}^{\bar{r}}\sum_{s=0}^{N-1} e^{-\frac{2\pi i}{N}(us+vr\sigma)} \\ &= \Delta(u)\sum_{r=0}^{\bar{r}} e^{-\frac{2\pi i}{N}vr\sigma} \\ &= \begin{matrix} \Delta(u).(\bar{r}+1) \text{ if } v = \text{integer} \times N/\sigma \\ \Delta(u) \text{ if } v \neq \text{integer} \times N/\sigma. \end{matrix} \end{aligned} \tag{13}$$

This result we recognize as a series of impulses along the v axis (figure 9D) with spacing N/$\sigma$. Thus axial periodic points with spacing (D) transform to periodic lines with spacing (N/D) , which in turn transform to axial periodic points with spacing (D). This shows that the periodic line pattern and the axial periodic impulse pattern are transform pairs (figures 7C &D; 9C & D).

**Evenly spaced point pattern.**
An evenly spaced pattern of points (figure 8 A & C) can be expressed as

$$\mathrm{H}(H,\theta) = \Delta(j-hH\cos\theta).\Delta(k-hH\sin\theta) \quad \text{where } h = 1, 2, .... \tag{14}$$

and the transform is

$$\mathrm{H}_T = \frac{1}{N}\sum_{h=1} e^{-2\pi\frac{iHh}{N}(u\cos\theta+v\sin\theta)}, \tag{15}$$

where $H$ is the distance between the points and $\theta$ is the angle(slope) of the point pattern. By analogy with the previous results for the impulse and the periodic line pattern this result can be identified as a set of lines with slope angle $\theta$ and distance between lines N/H (see figures 8B & D).

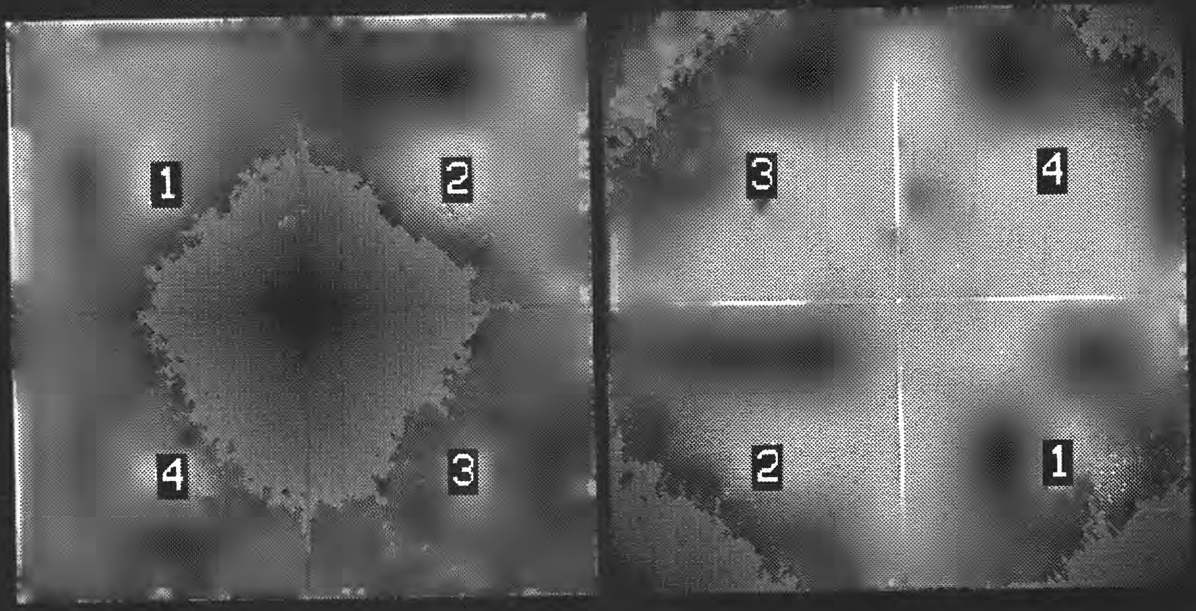


Fig. 1 - Comparison between the FFT 2D transform (left) and the optical transform (right). The dc term(zero frequency) is the single bright pixel (upper left for the Fourier transform, center for the optical transform). Quadrant numbers are provided to show which quadrants correlate in the two transform displays.

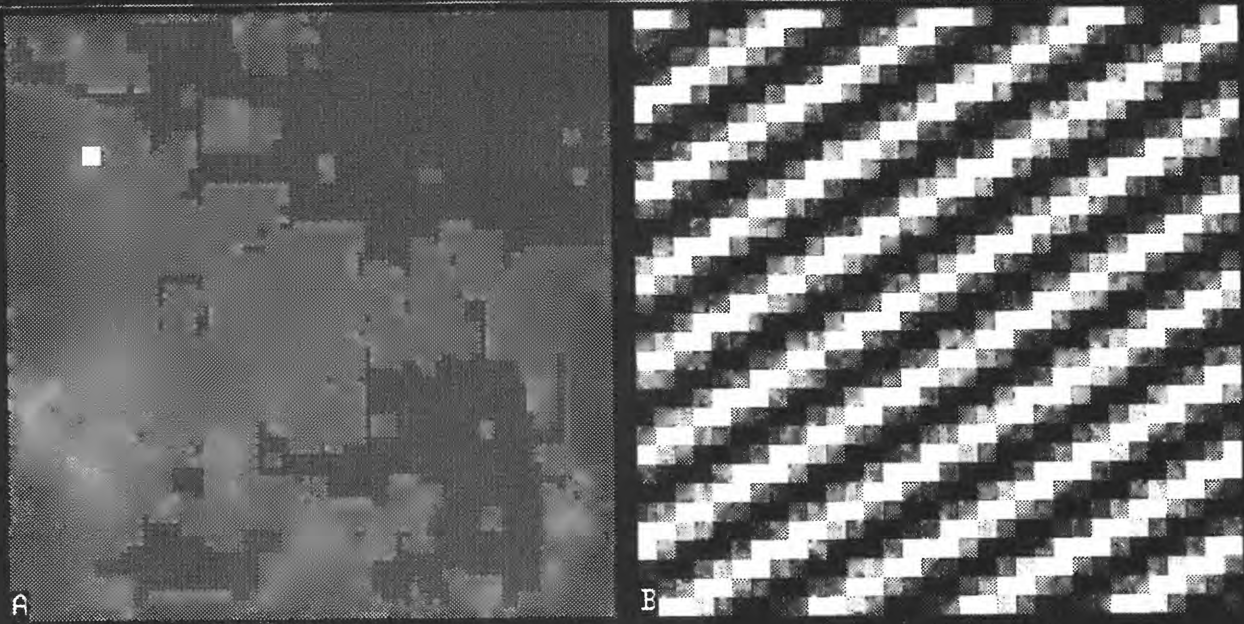
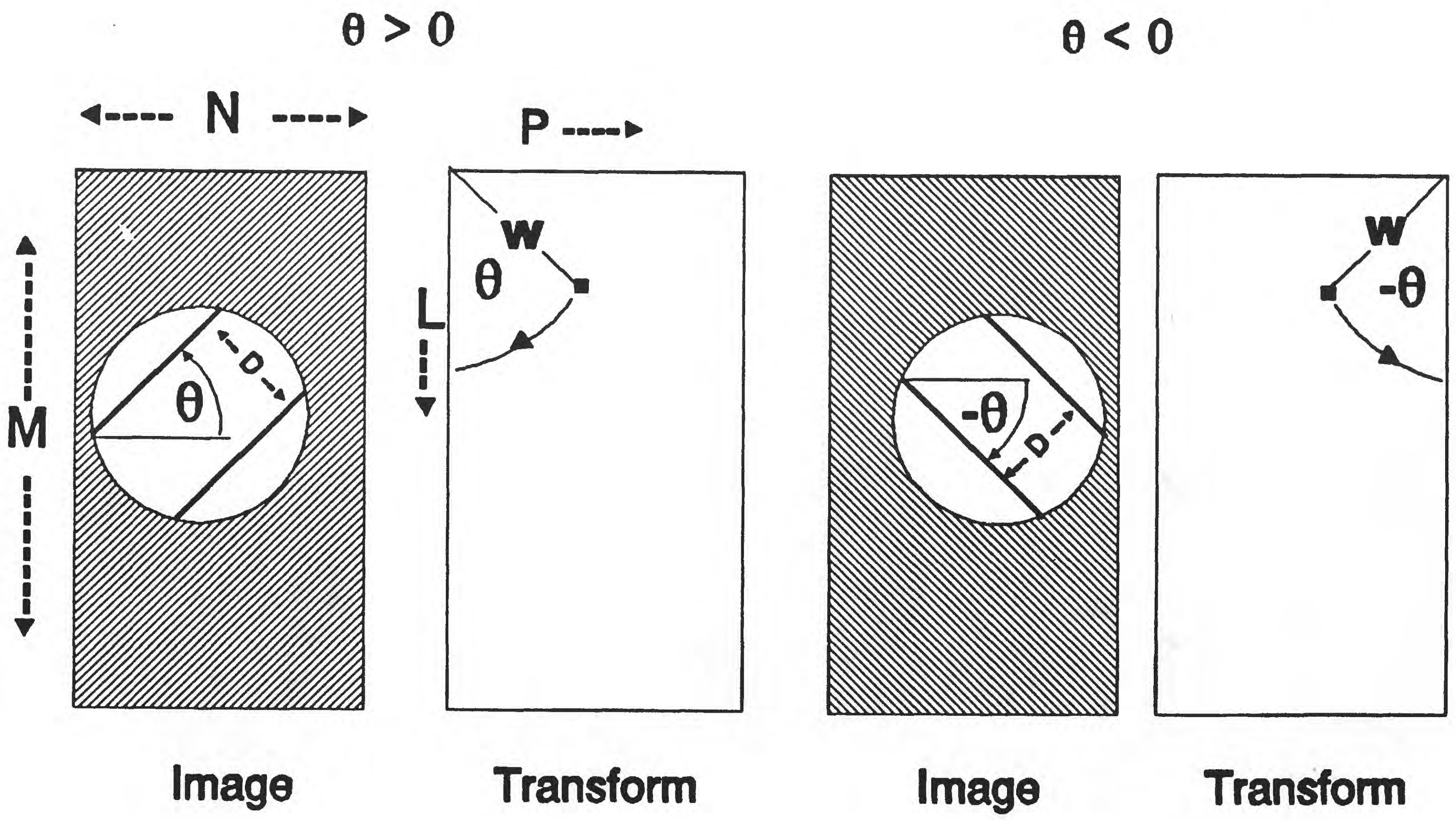


Fig. 2 - An image of a simple impulse (A) and the frequency image of its transform(B).

Fig. 3 - Parameters used to describe the geometric relationships between a pattern (periodicity D, slope $\theta$ ) in an N by M image and its transform impulse at (P,L). The second pair shows the relationship for $\theta < 0$. The radial length W is computed in equation 3.



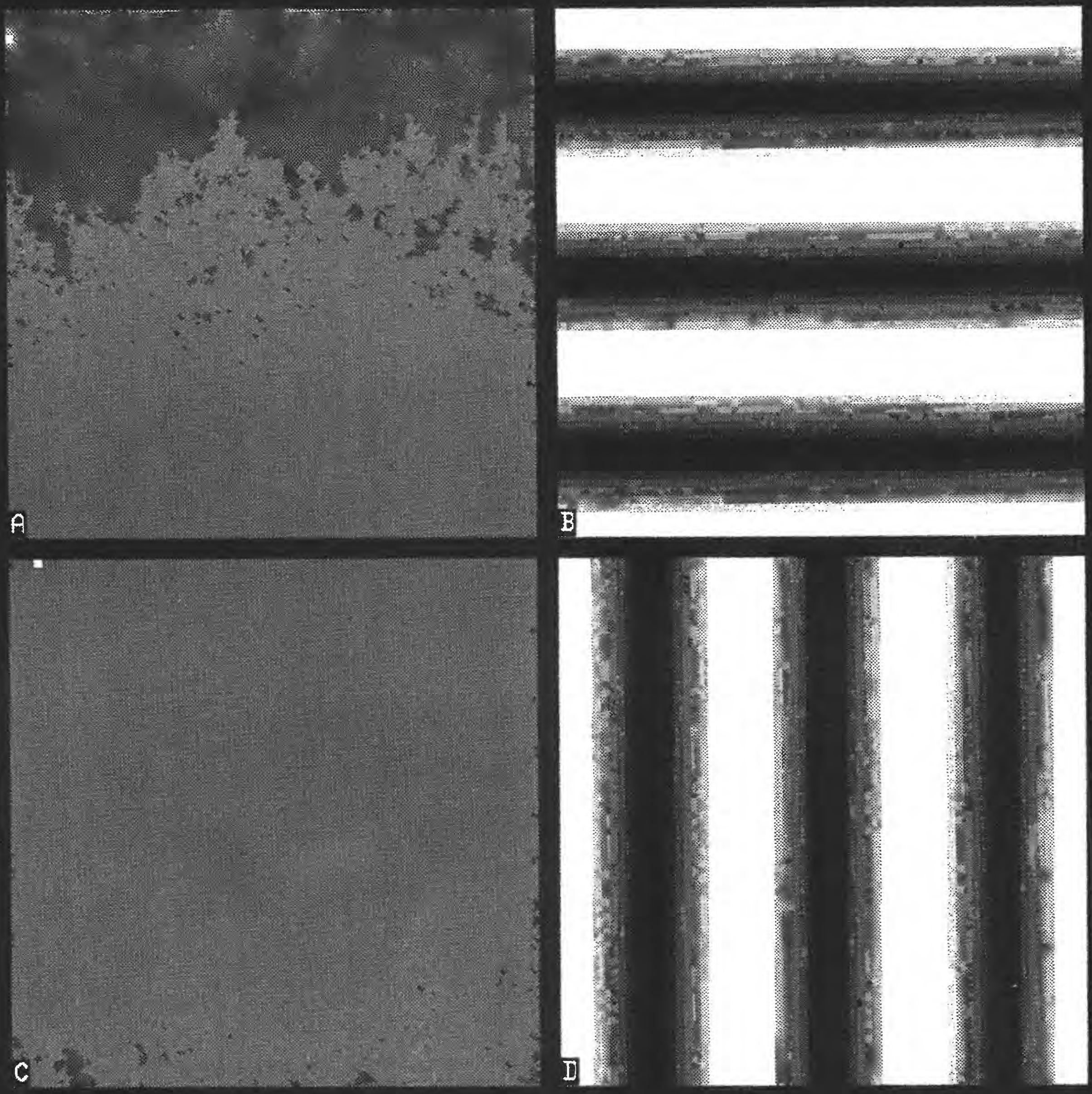


Fig. 4 - Image and transform pairs for an impulse on the line (A) and the pixel (C) axis. Note that the transform of a horizontal pattern(B) lies on the vertical(line) axis(A) and vice versa(D and C).

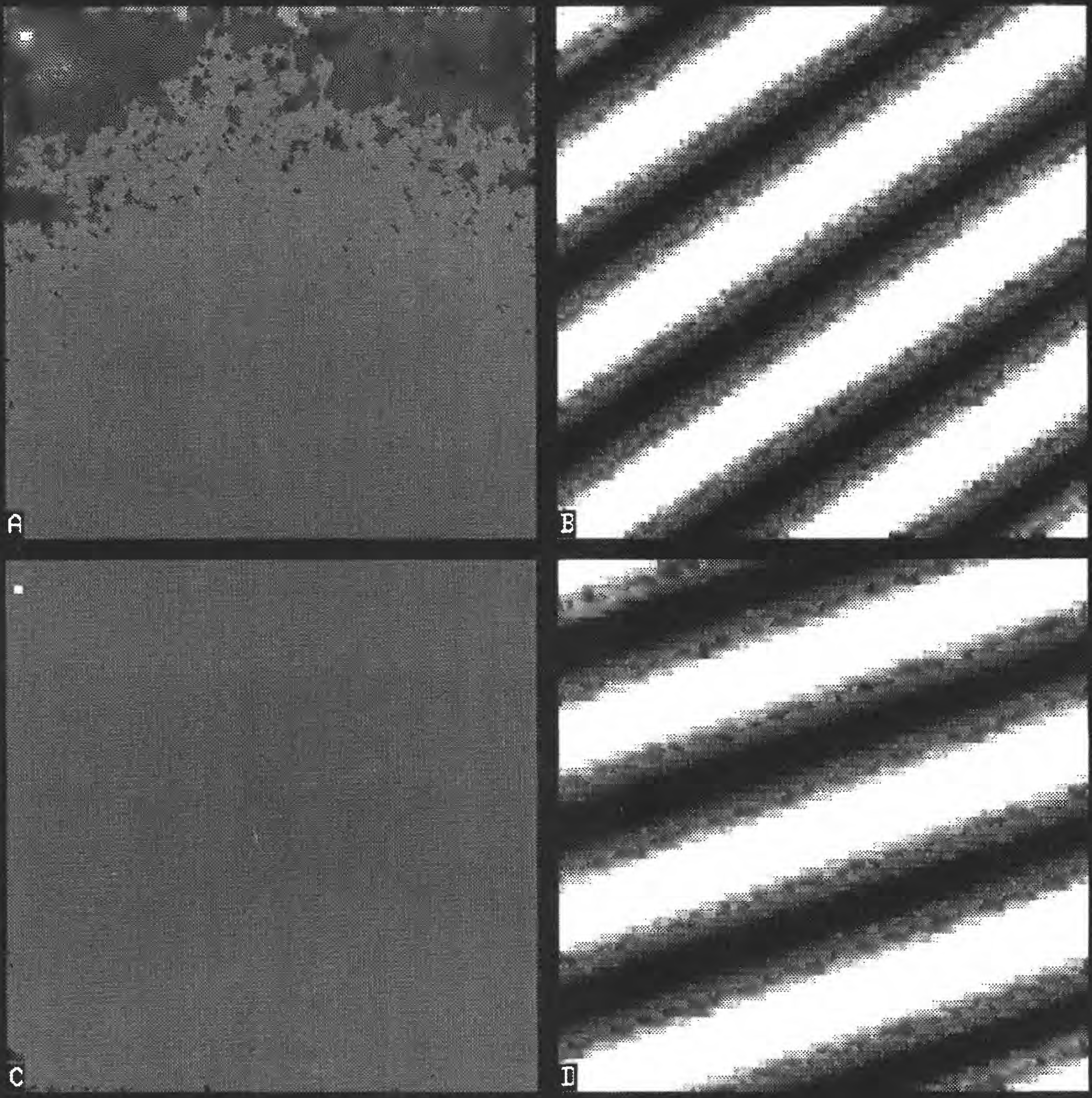


Fig. 5 - Image and transform pairs for an off axis impulse. Note that the angle of the transform pattern changes with the slope of the line drawn from the origin to the impulse.

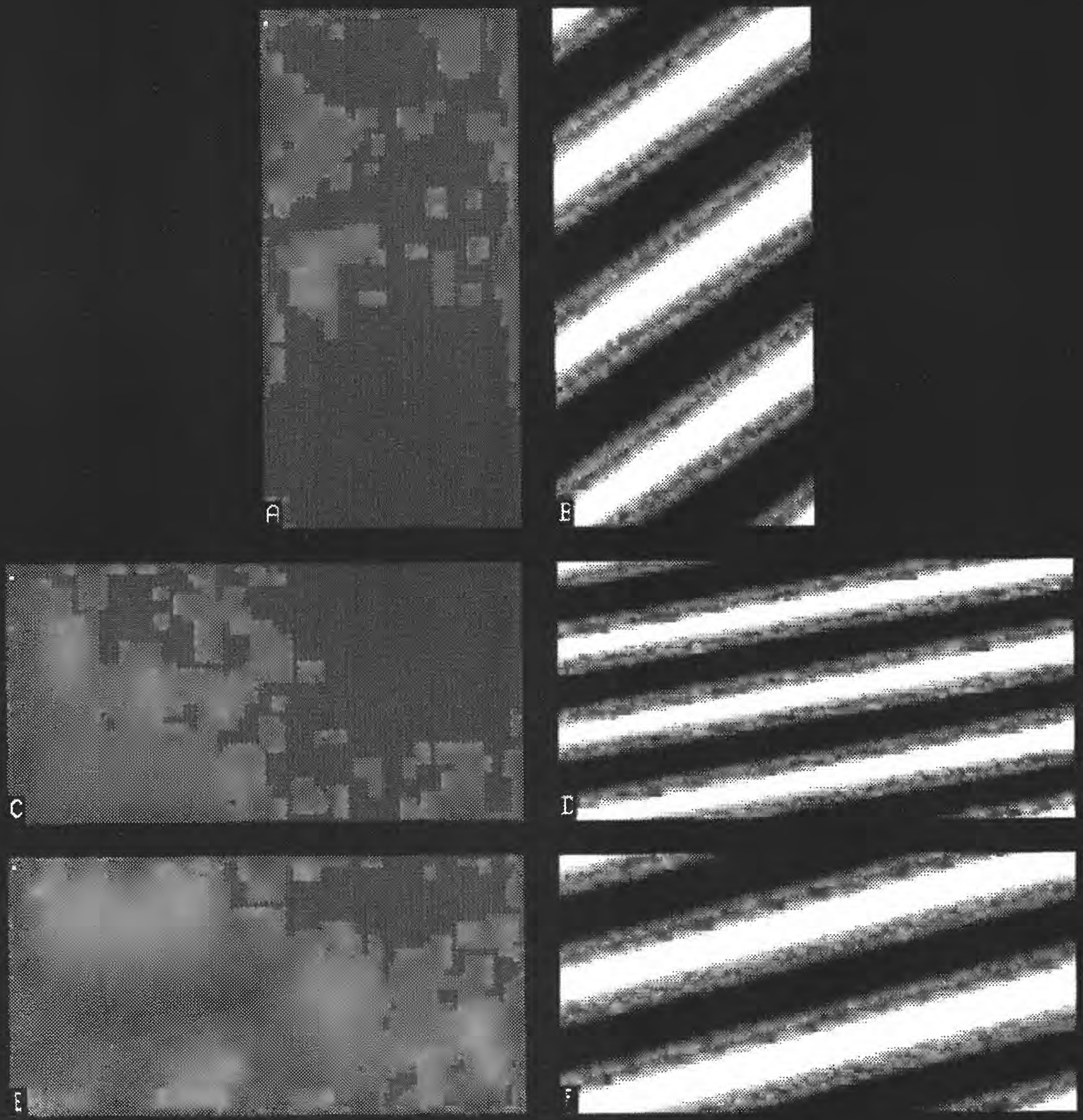


Fig. 6 - Image and transform pairs for different size images. Note that although the impulse is in the same location in images A and C the pattern of the transform changes because the image size has changed. In E the position of the impulse was changed to yield a transform pattern in F similar to that in B. The pattern is not identical due to integer arithmetic (see text).

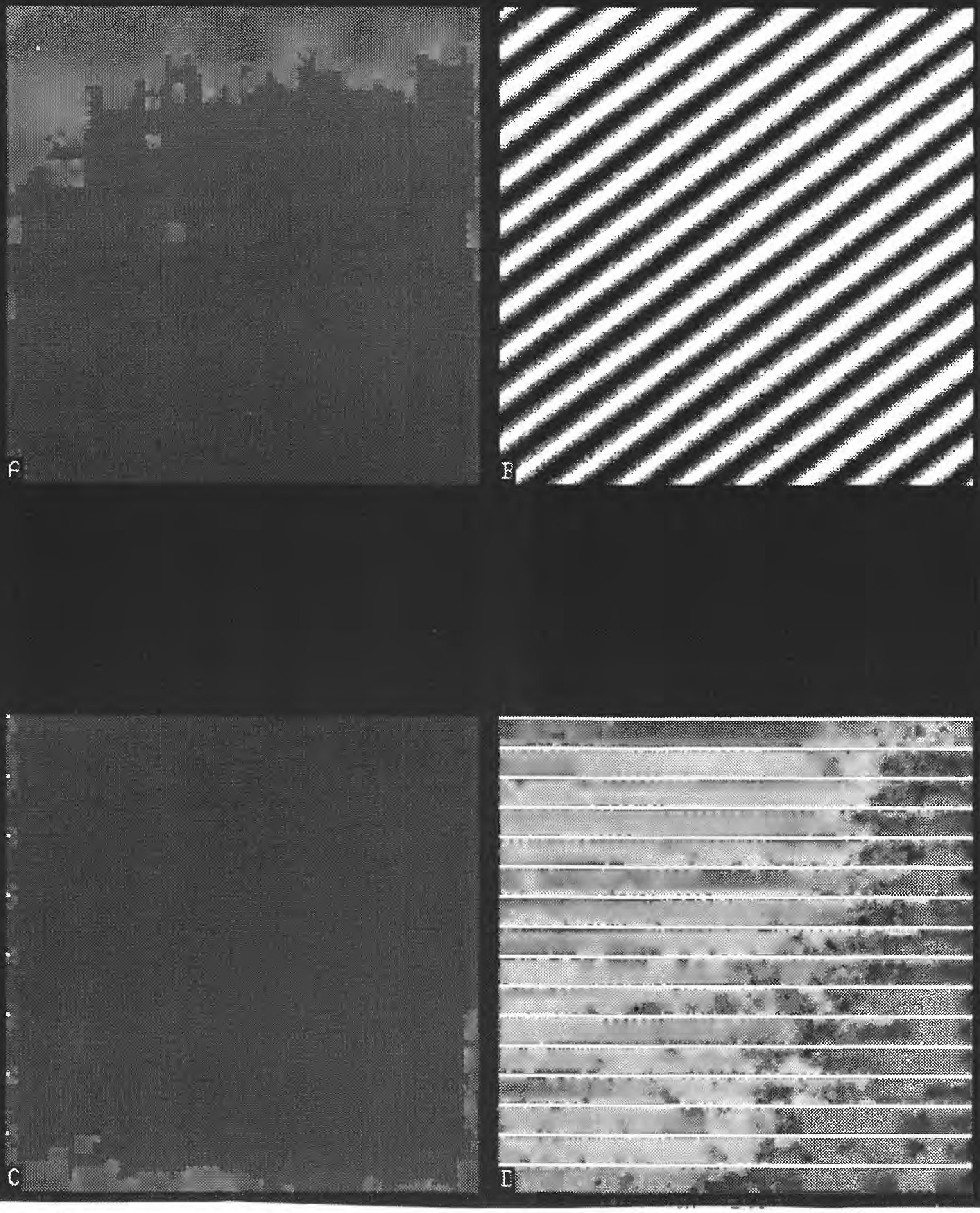


Fig. 7 - The upper panels (A, B) show an image transform pair for a single impulse. The bottom panels (C, D) show the image transform pairs for a periodic impulse along the vertical (line) axis. Note that the single impulse has a corrugated transform pattern, whereas the periodic impulse has a linear transform pattern of high(bright) values like rows of "walls".

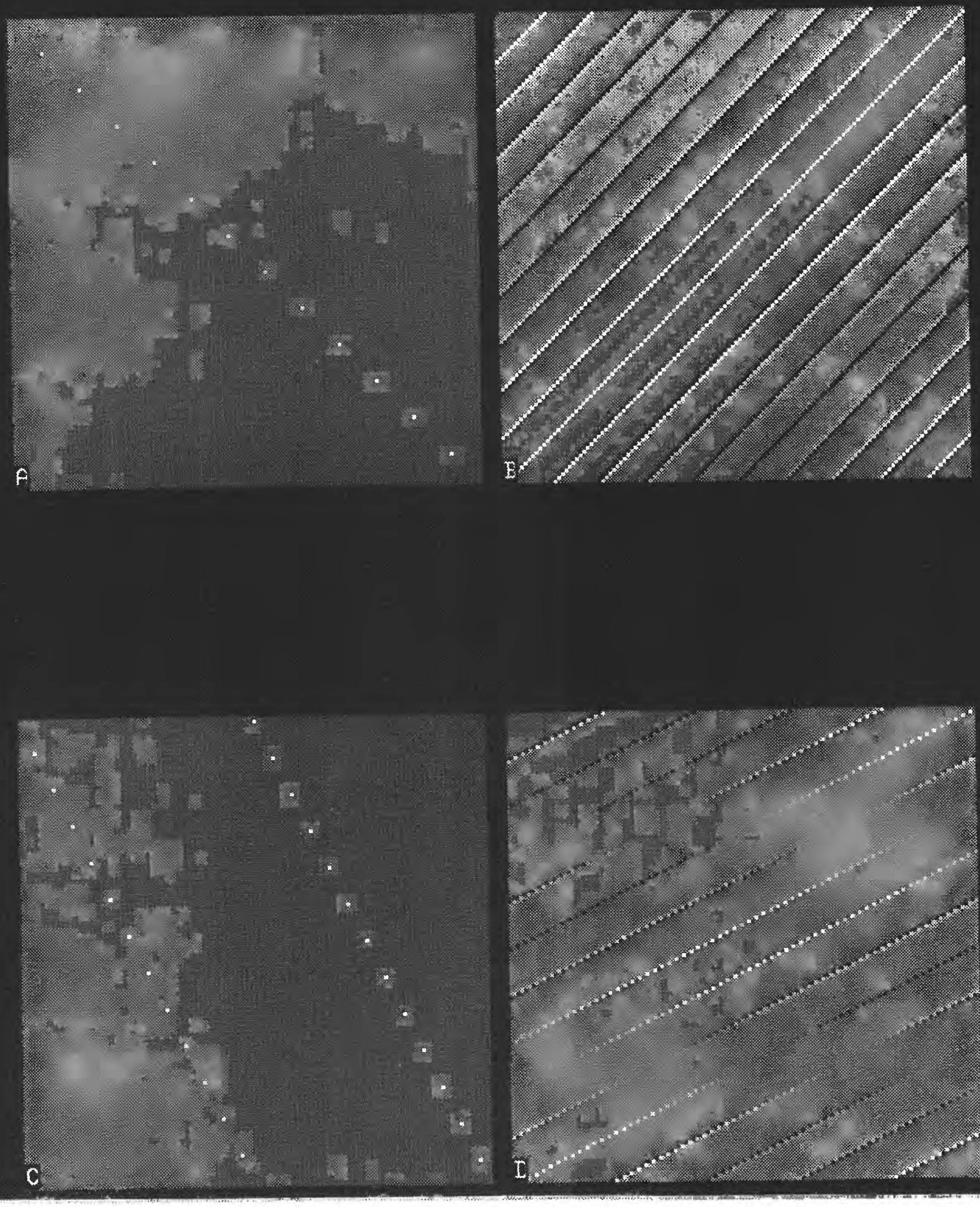


Fig. 8 - The results shown in the figure 7 (C,D) are generalized for an inclined periodic impulse pattern. The rotation of the transform pattern is related to the angle(slope) of the line of the periodic impulses. Note that in panel C the periodicity of the function results in a wrapping around from the bottom of the image to the top. This implies that the transform of a noise pattern similar to panel D will exhibit the same wrap around shown in C if enough harmonics are present.

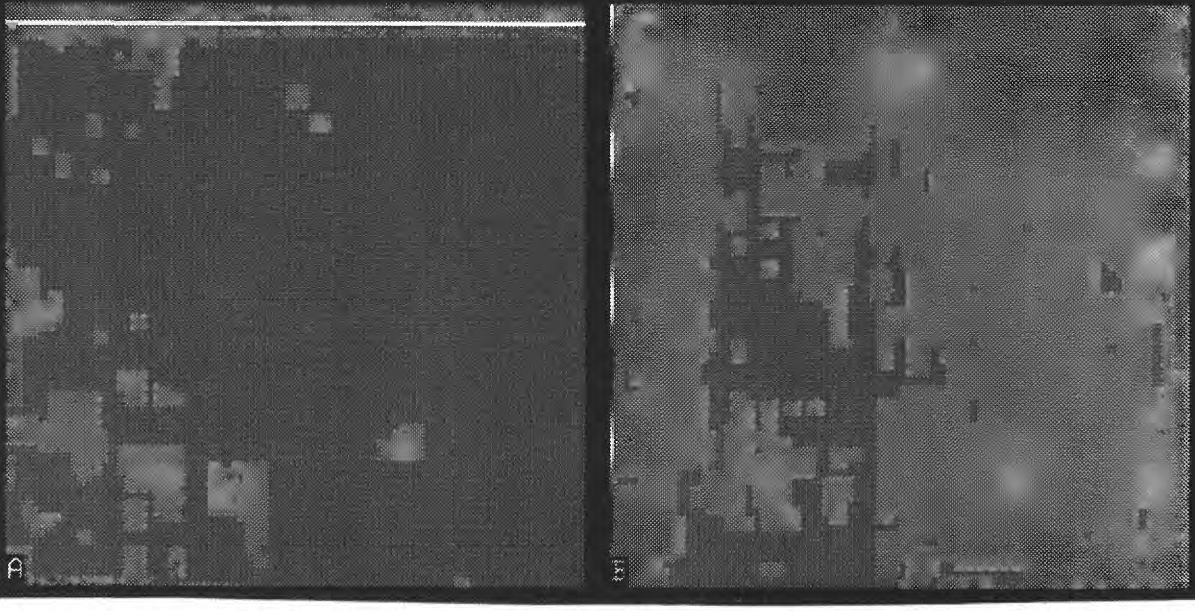
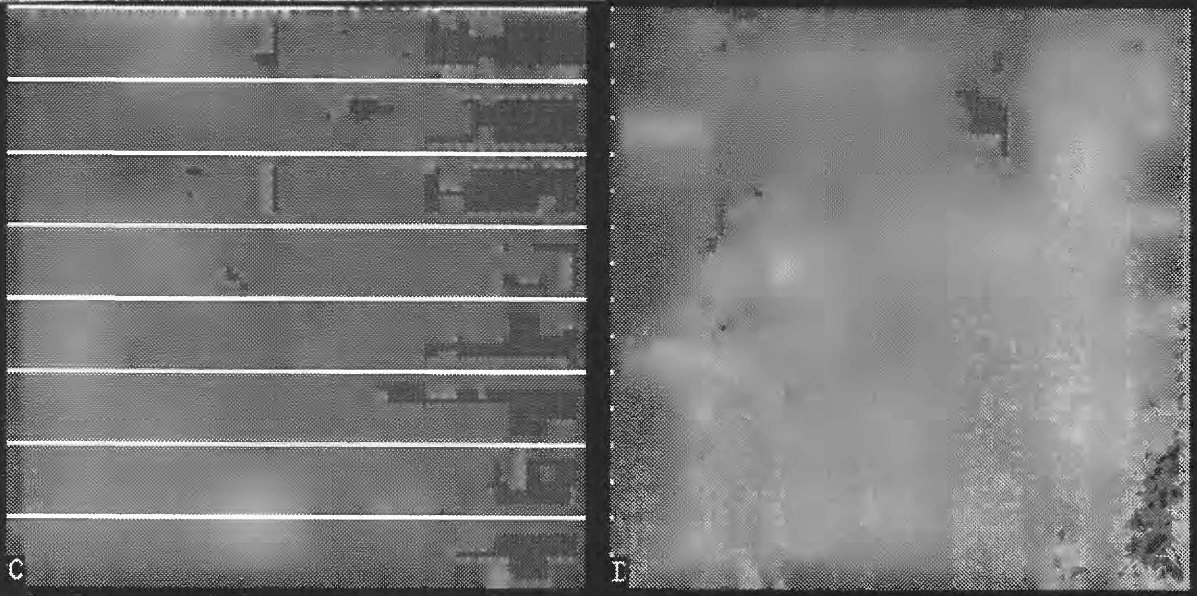


Fig. 9 - A line in image space is just a collection of adjacent impulses. The upper panels (A, B) show the transform of a single horizontal line. Note that the corrugated pattern occurs only along the vertical axis of the transform. The bottom panels (C, D) show the transform for a set of periodic horizontal lines. The transform pattern also occurs only along the vertical axis and the corrugated pattern has been compressed into a periodic set of spikes. This result is similar to the previous case for an impulse and a periodic impulse (Fig. 7). Note that as the pattern spreads out in image space, its transform contracts (and vice versa). Panels C and D also illustrate a well-known property of the 2D FFT that a periodic "wall" pattern and a periodic array of points are transform pairs of each other.



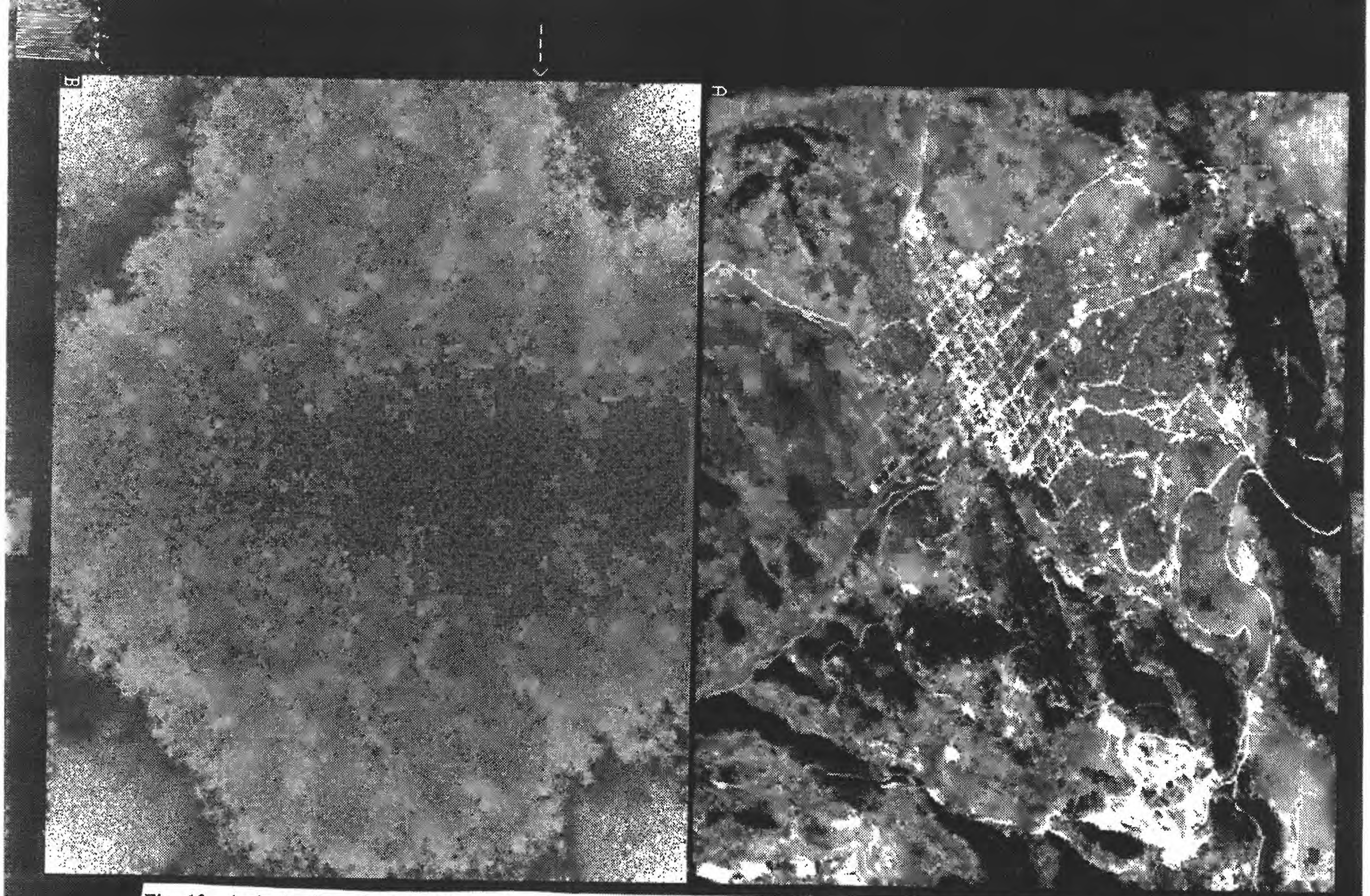


Fig. 10 - A single channel of an airborne imaging spectrometer (GERIS) data set of an area covering Cripple Creek, Colorado and its transform. Close examination of the image (A) reveals scanline noise which appears as light horizontal lines across the image. These lines are parallel to the direction the image is scanned. In the transform (B), this noise appears as a bright pixel and is indicated with an arrow. (Because of the transform symmetry a similar bright pixel can also be seen at the same distance up from the bottom of the transform. Between these two bright pixels is a less intense pixel caused by a higher harmonic of the noise.)

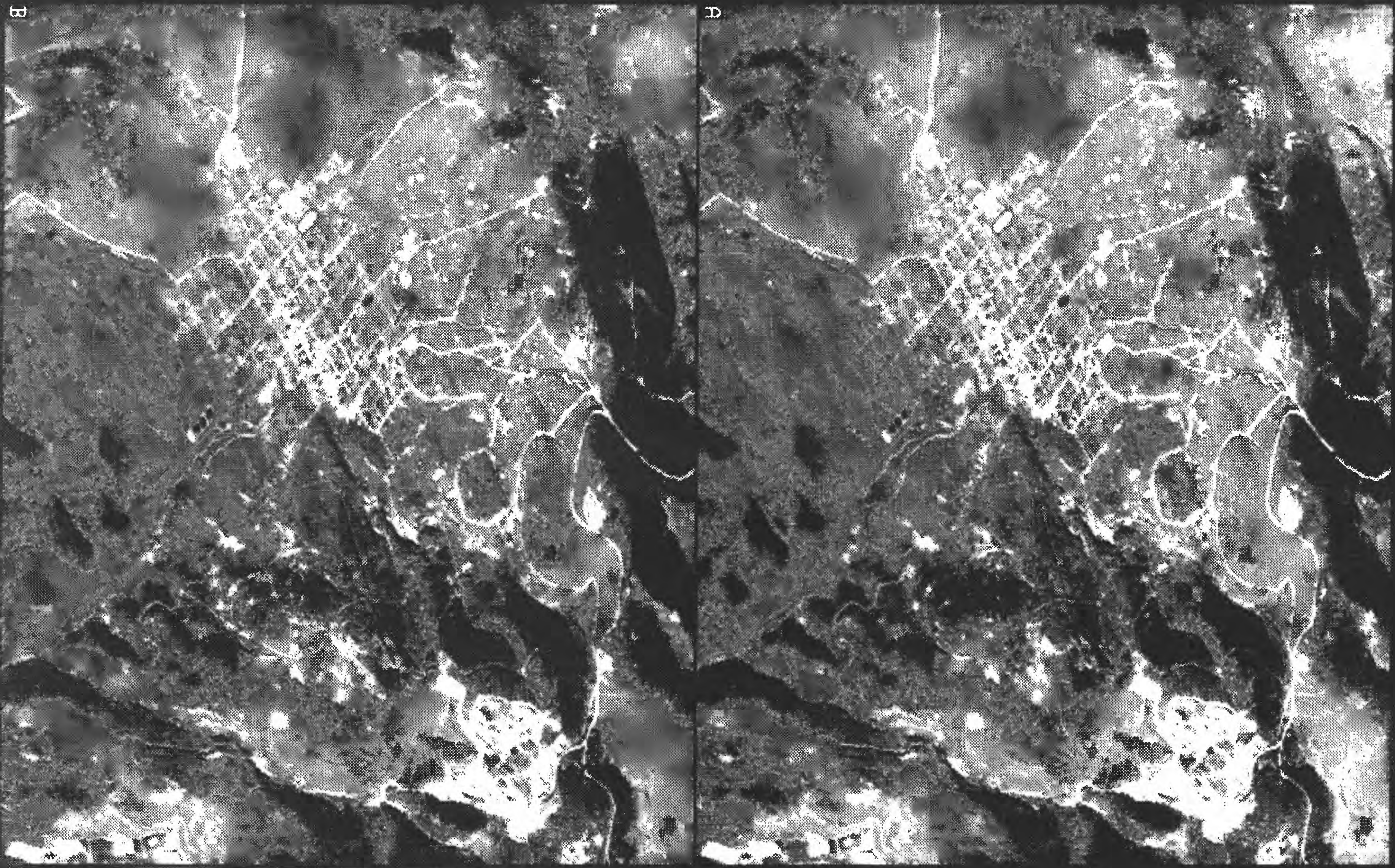


Fig. 11 - A blocking filter was applied to the noise spike shown in the previous figure (10 B); the inverse transform is shown in panel A. Although the scanline noise has been reduced, there is some still present. Examination of the transform (fig. 10B) shows an additional periodic component present as noted in figure 10. A second blocking filter was added and the results shown in figure 11B.

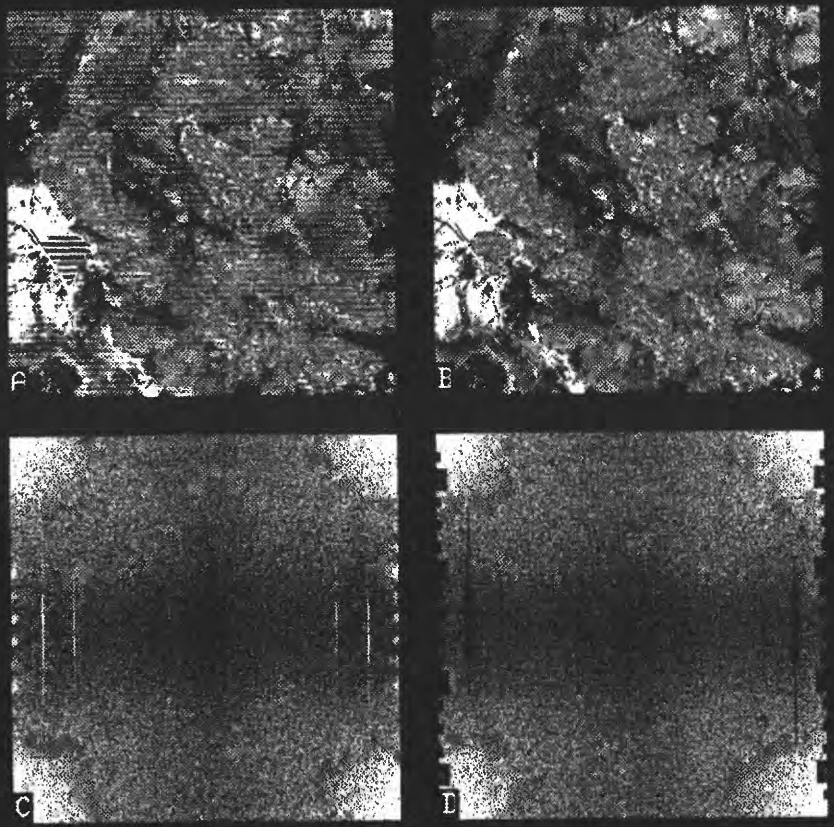


Fig. 12 - A band ratio image of TM satellite data of Bolivia (12A) shows a very distinctive noise pattern in its transform (12C). The pattern is periodic so a periodic box filter was applied along the line axis to remove this noise. (Additional supplementary filters were applied to the vertical line segment noise patterns. This type of filter will be discussed in conjunction with figure 16).

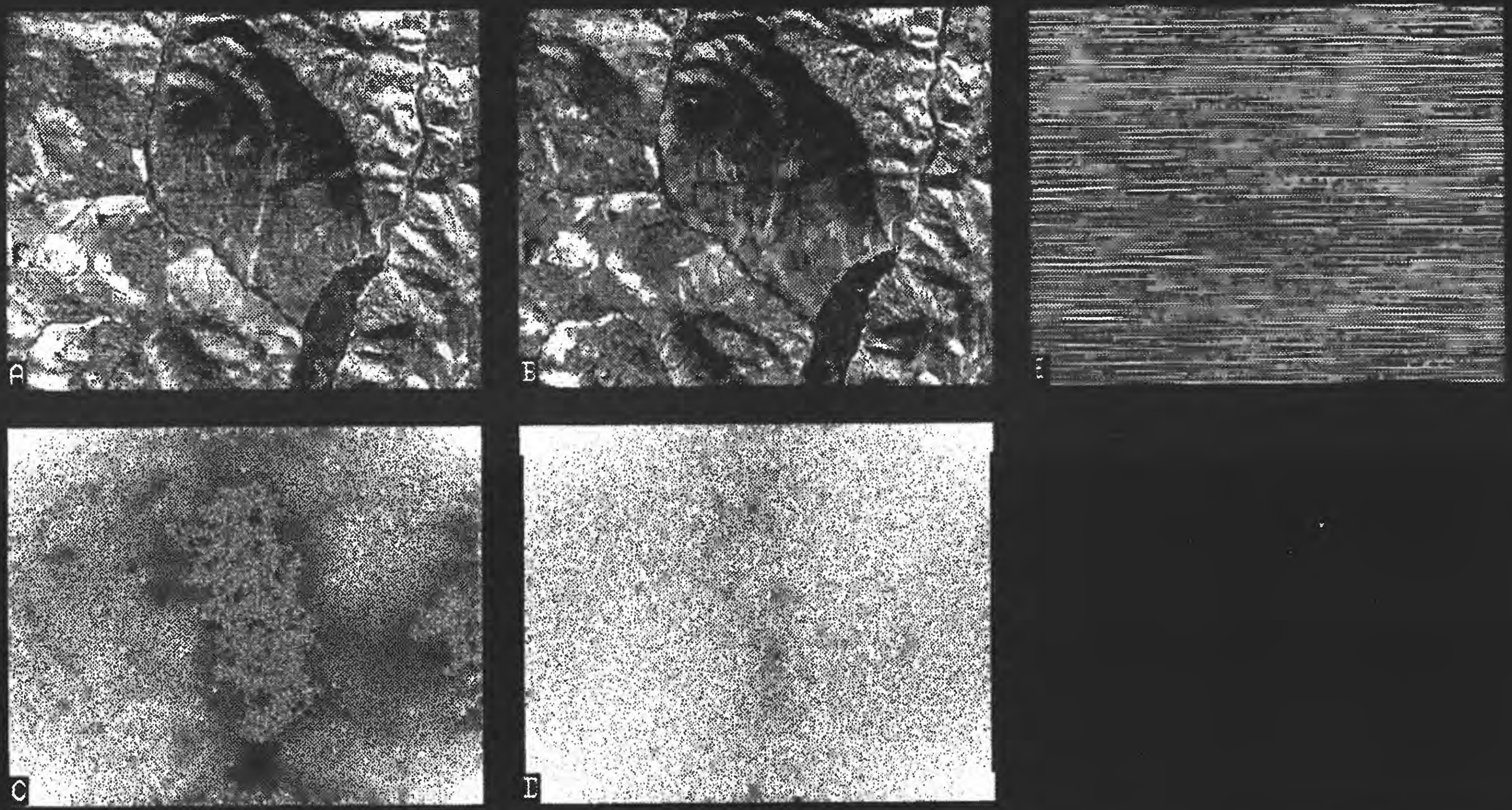


Fig. 13 - A second principle component image of TIMS aircraft data for Iron Hill, Colorado (13A). Because the scanline noise is not entirely periodic, a 'u-shaped' filter was applied along the line axis of the transform. (It can be seen in the transform image (12D) as a black indentation on the left hand side). The filtered image (12B) shows the filtering improvement, and the difference between the filtered and unfiltered images (12E) shows the nature of the removed noise.

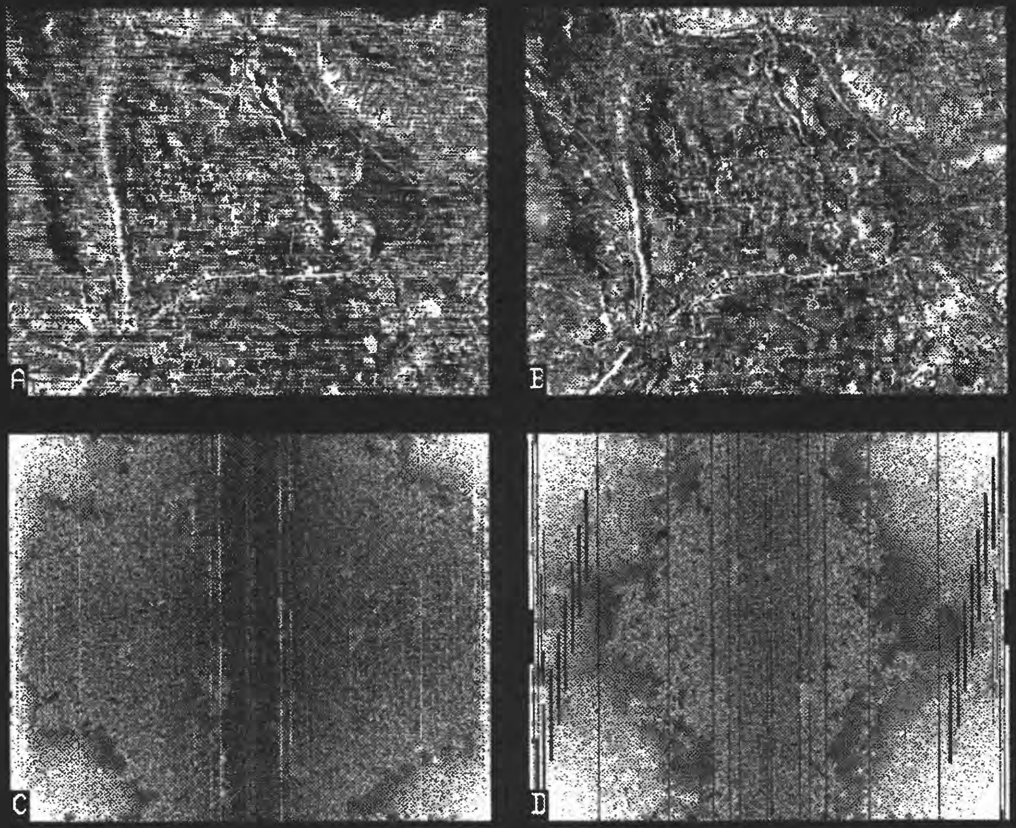


Fig. 14 - TIMS aircraft data for the Canon City, Colorado area (14A). Although a more complex noise pattern is evident (14C), a variety of filter shapes can be directly applied (14D) and the resulting image is shown (14B).

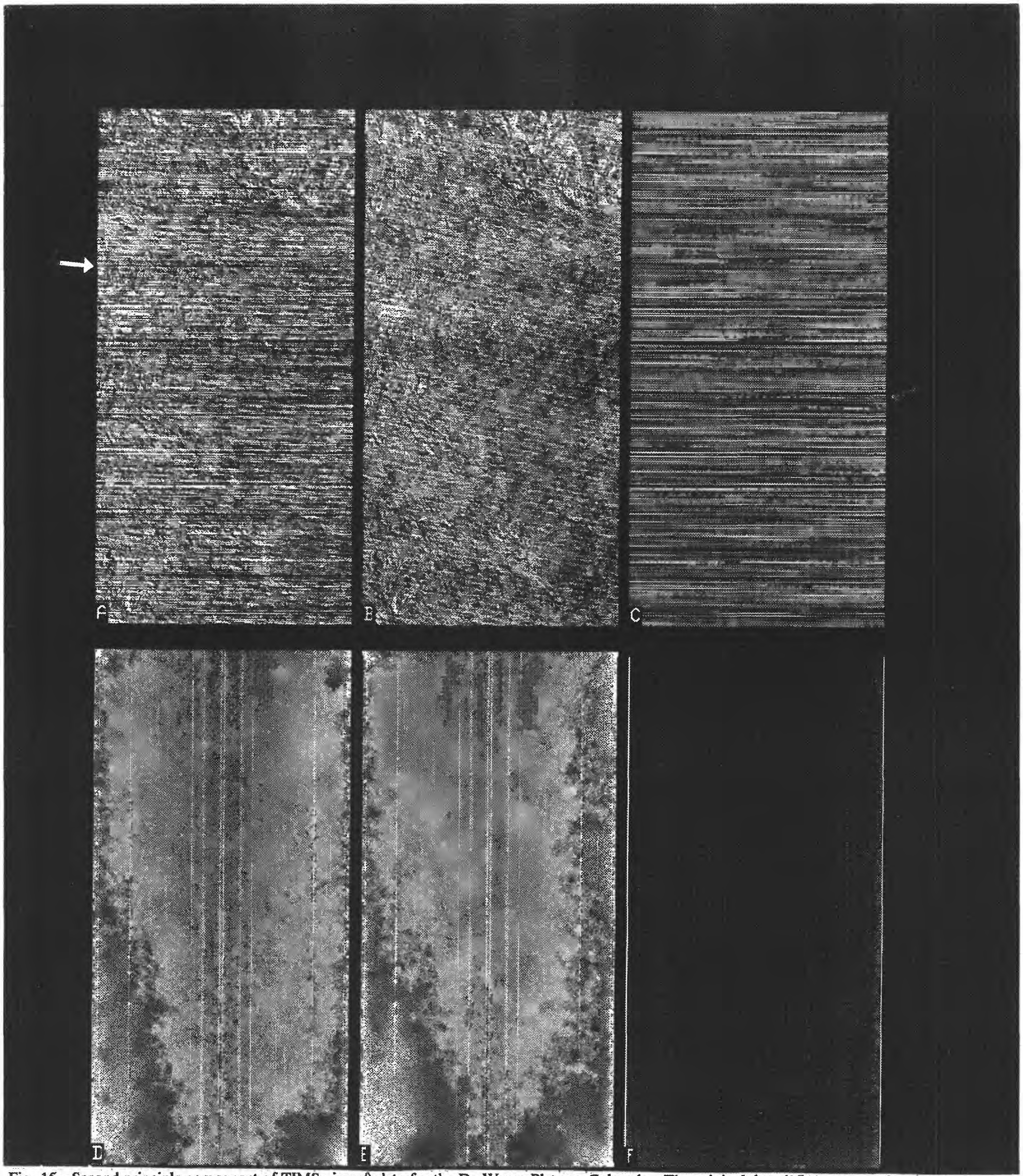


Fig. 15 - Second principle component of TIMS aircraft data for the De Weese Plateau, Colorado. The original data (15A) are so noisy that theyt appears unusable. A 'u-shaped' filter was applied to reduce the horizontal scanline noise (see arrow). Multiple noise patterns are present in these data that become evident as more prominent noise is removed. In the next five following figures, filters will be applied successively to remove different types of noise. Figure 21 provides a summary comparison of the total filtering process. Each figure consists of 6 panels. The left panels are the image (A) and its accompanying transform (D). The center panels are the filtered image (B) and its transform (E). The right panels are the difference between the images (C) and the transforms (F). Panel C shows the noise that is removed by the filter seen in panel F as bright. To assist in the comparison the left panels in each succeeding figure is identical to the center panels(filtered) in the previous figure.

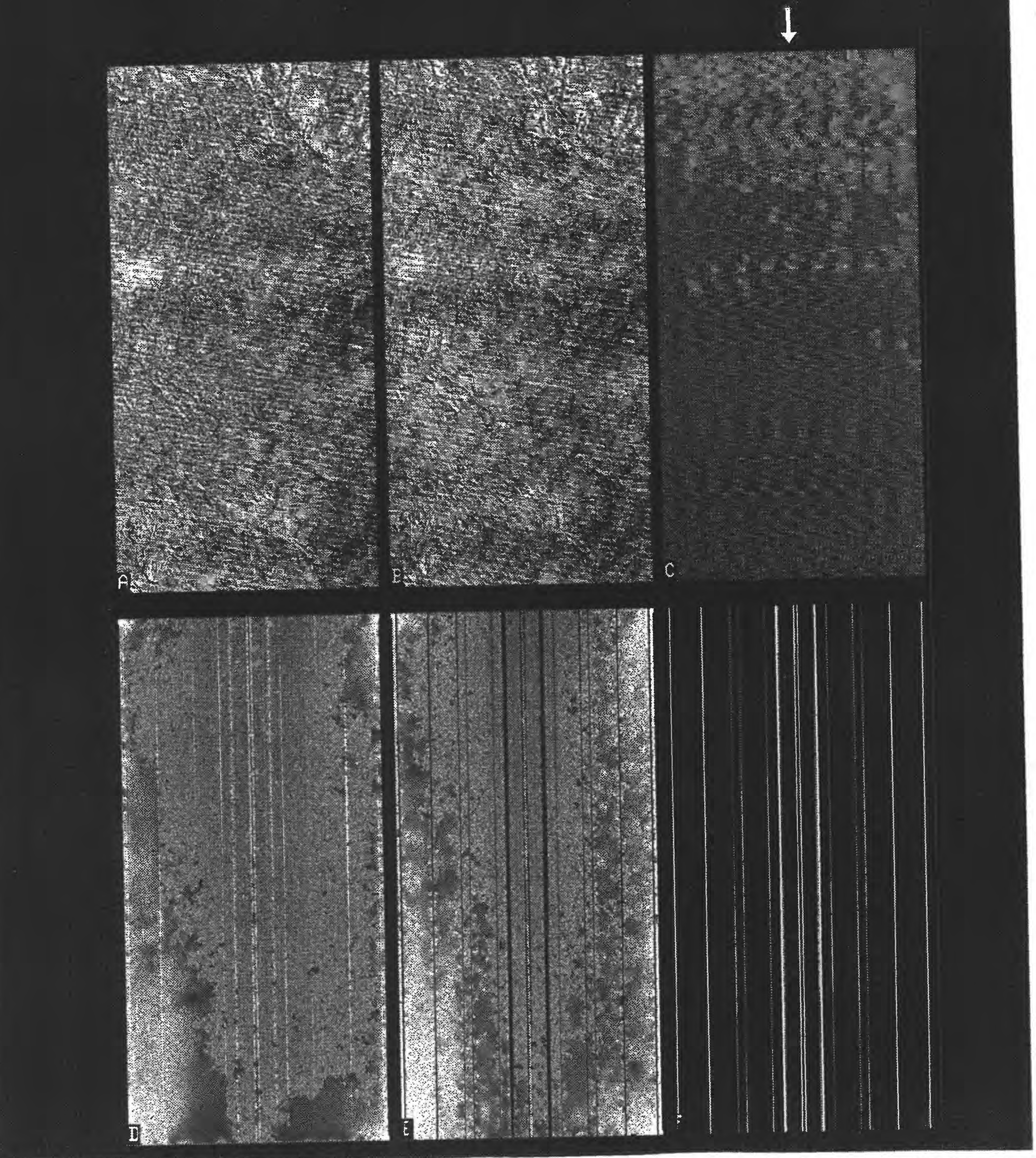


Fig. 16 - Full vertical line filters (dark vertical lines in E) applied to remove noise apparent in the transform (bright vertical lines in D). The filtered noise is shown in C and the arrow indicates the primary direction of the nosie pattern. See figure 15 caption for a more complete explanation.

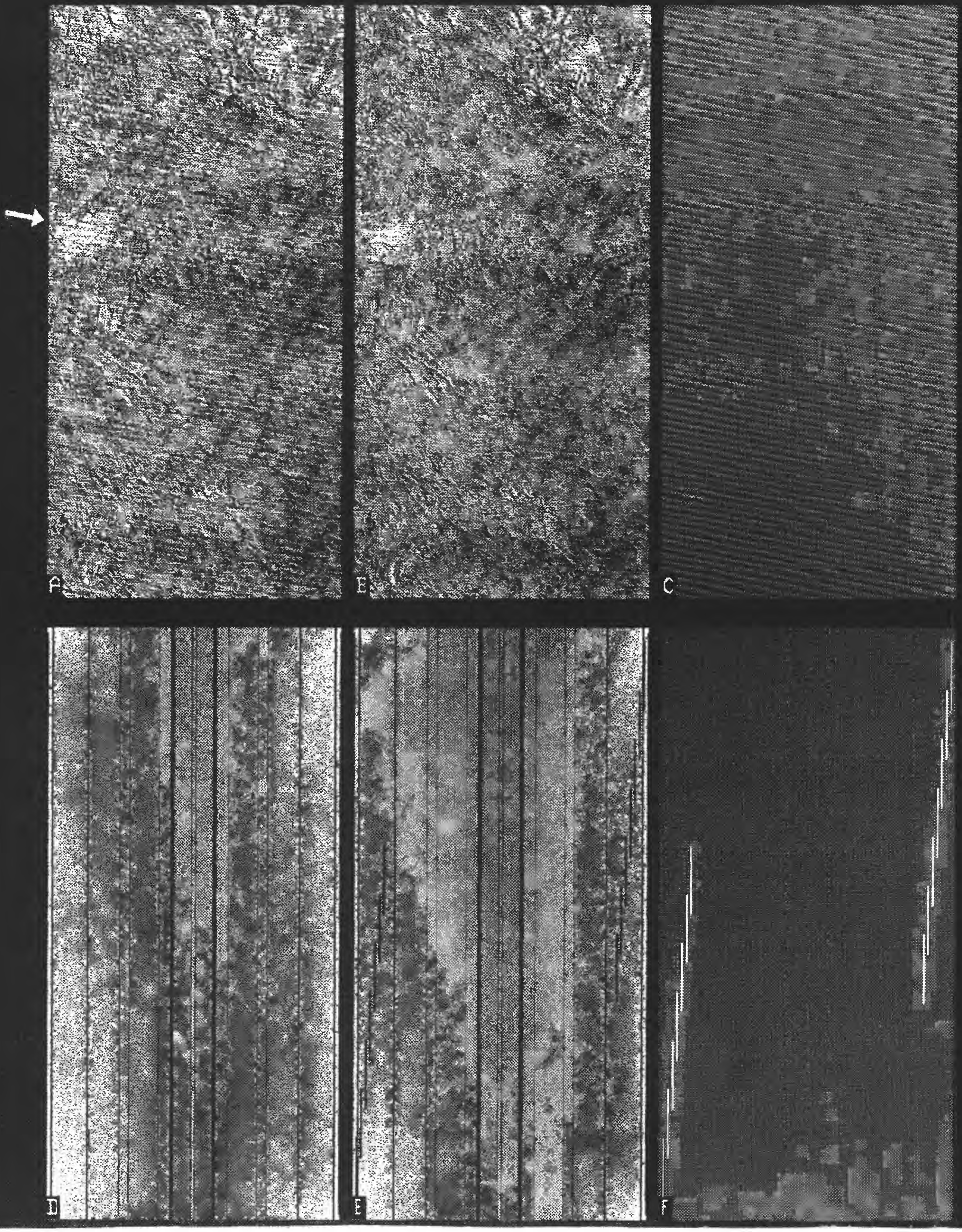
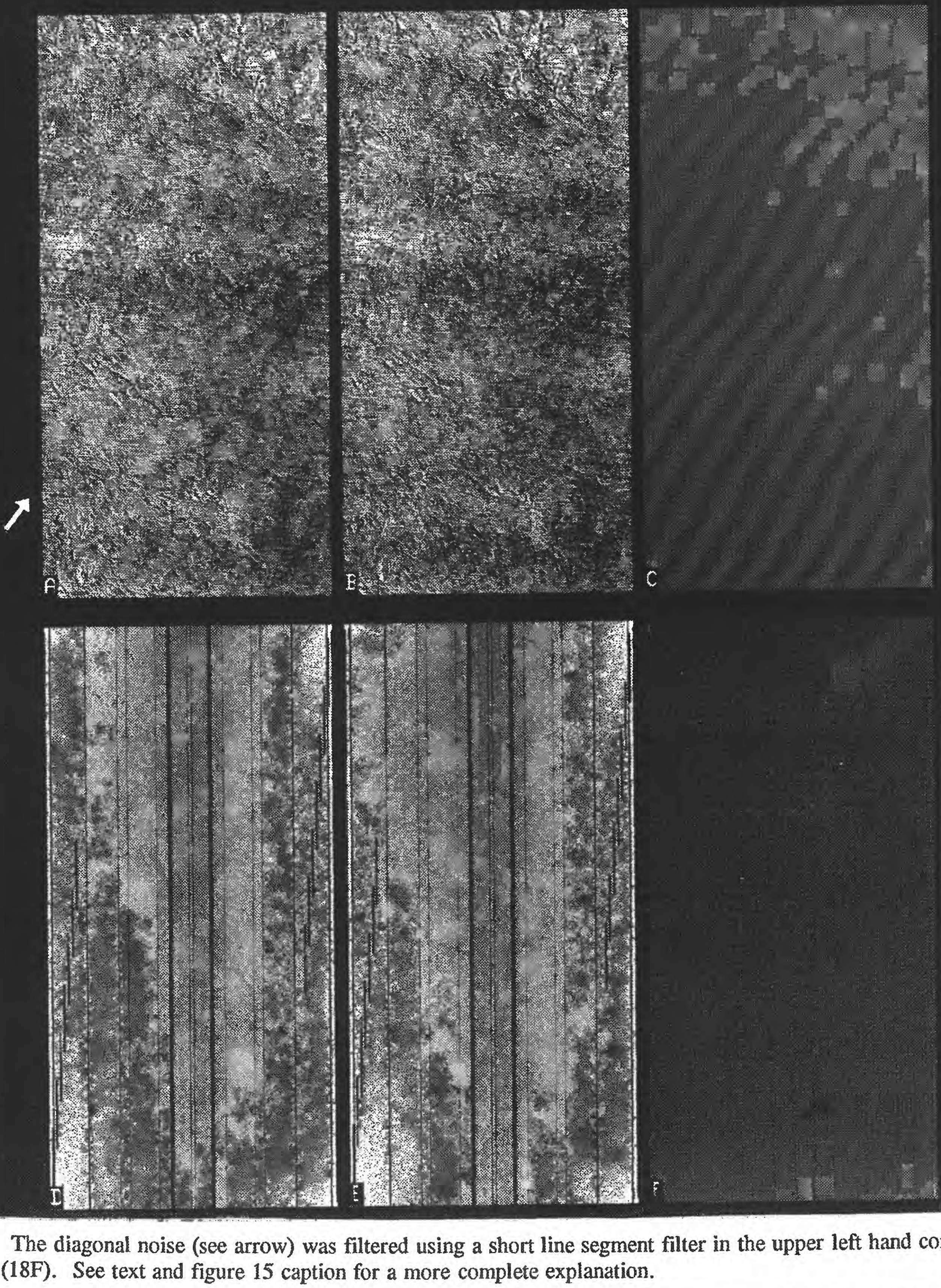


Fig. 17 - An angular harmonic filter. See text and figure 15 caption for a more complete explanation. The arrow indicates the direction of the noise being filtered.



'ig. 18 - The diagonal noise (see arrow) was filtered using a short line segment filter in the upper left hand corner of the 'ansform (18F). See text and figure 15 caption for a more complete explanation.

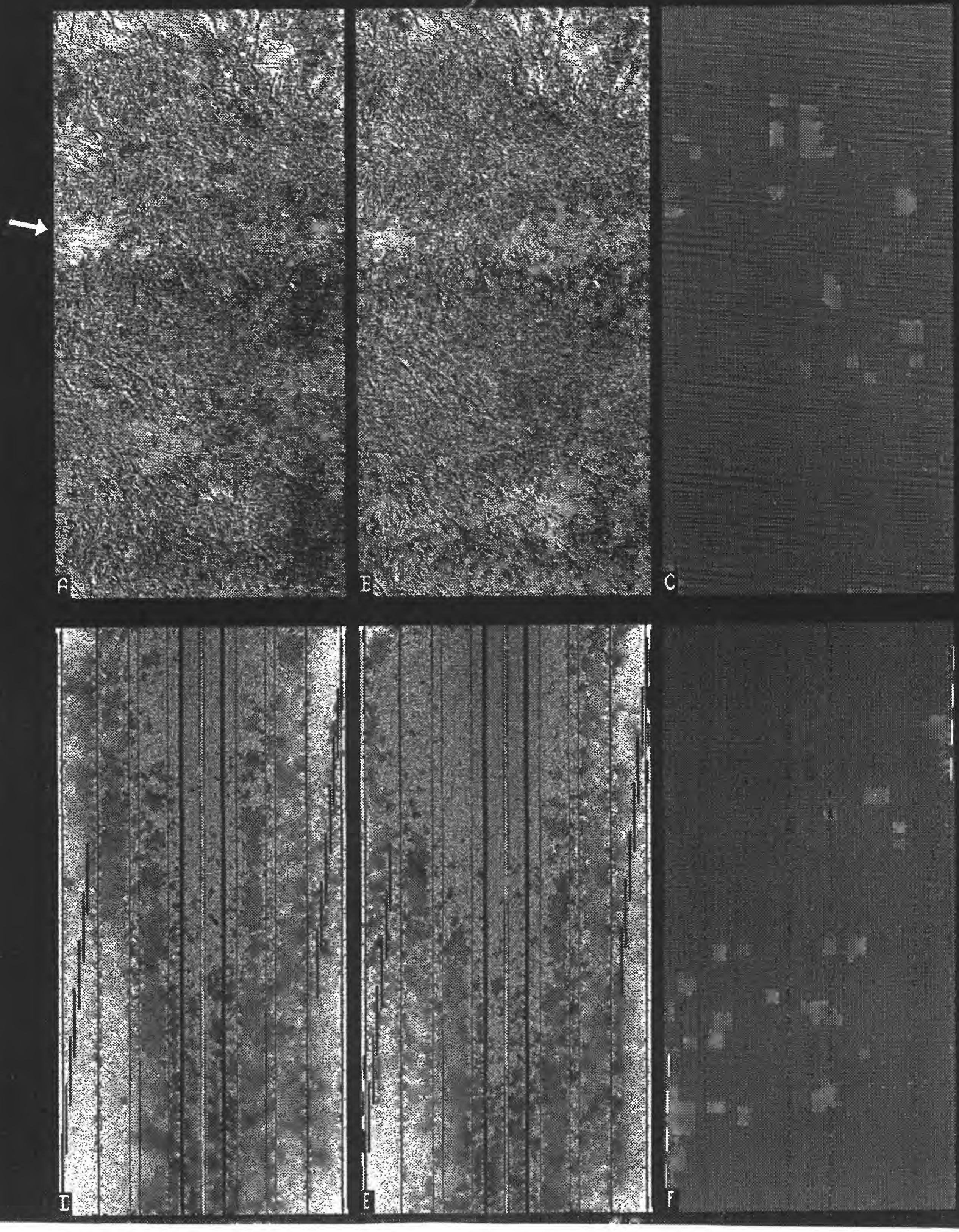


Fig. 19 - A second angular harmonic filter was then applied. See text and figure 15 caption for a more complete explanation. The arrow indicates the orientation of the noise pattern.

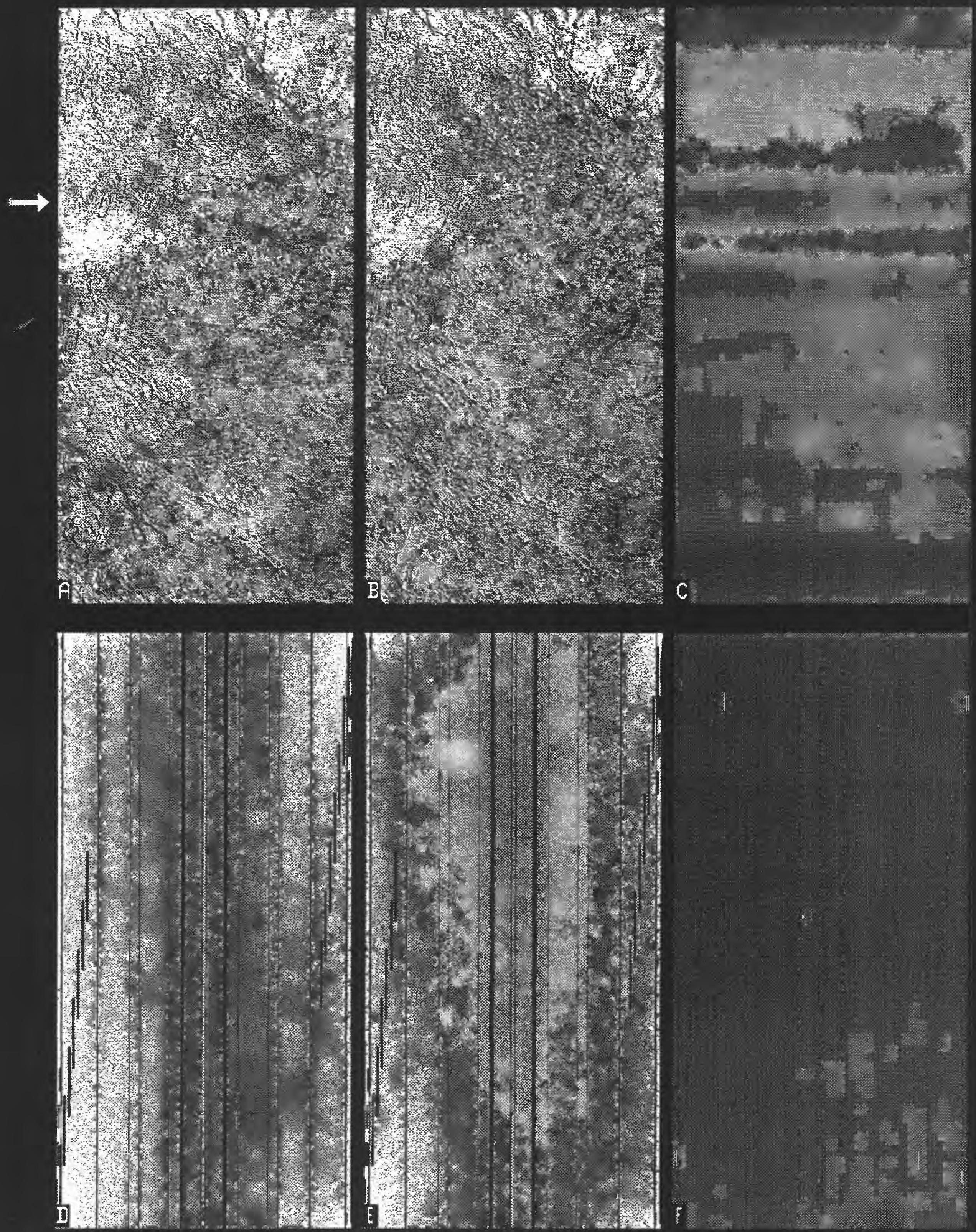
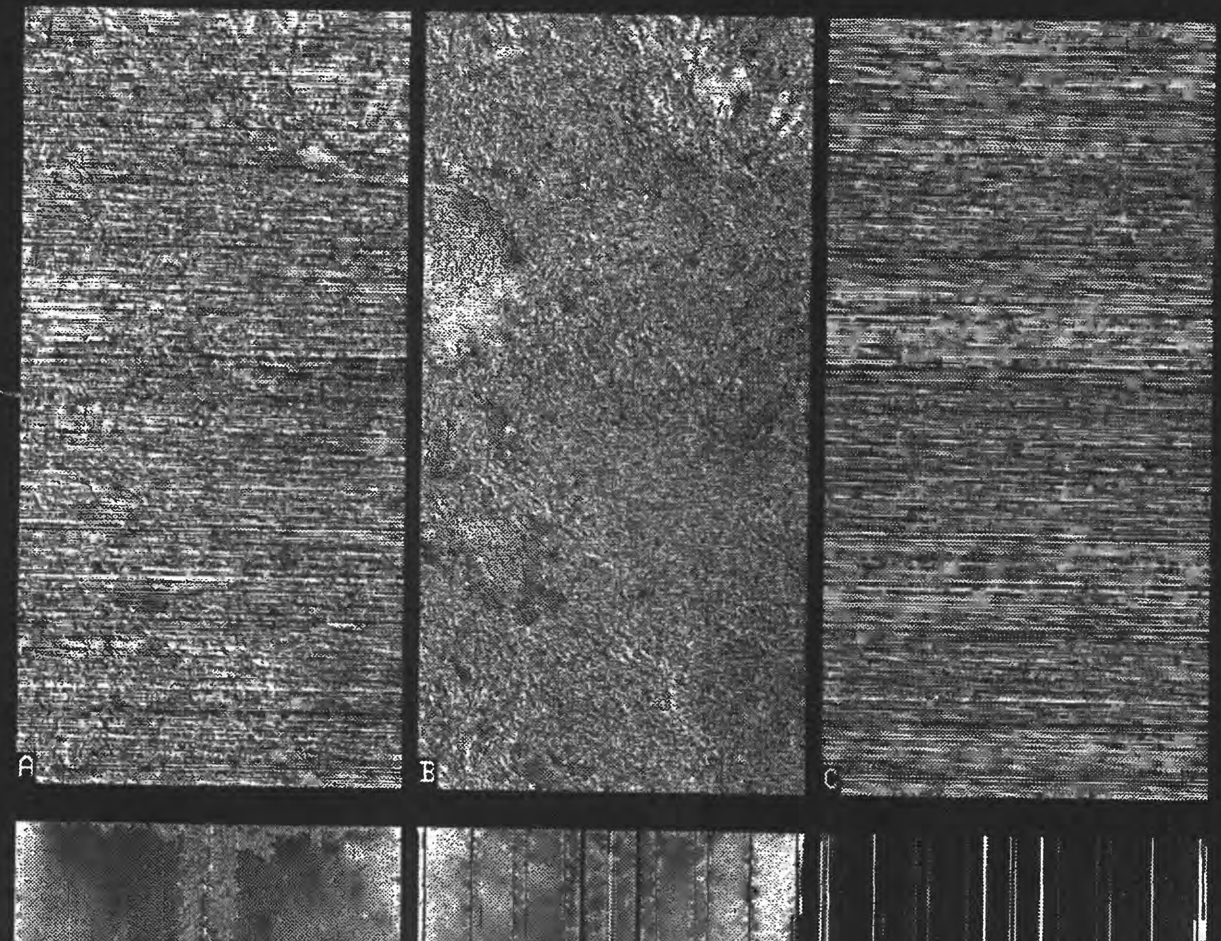


Fig. 20 - The last set of filters applied were four short line segments evident in the corners of the transform (20F). The primary noise removed is horizontal (see arrow) and low frequency (C).

Fig. 21 - A composite of the filters applied in figures 15 through 20. The original image (A) can be compared with the final filtered version (B) and the removed noise (C). The filter patterns are evident in the transform (dark lines on E) and transform difference (bright lines on F) images.



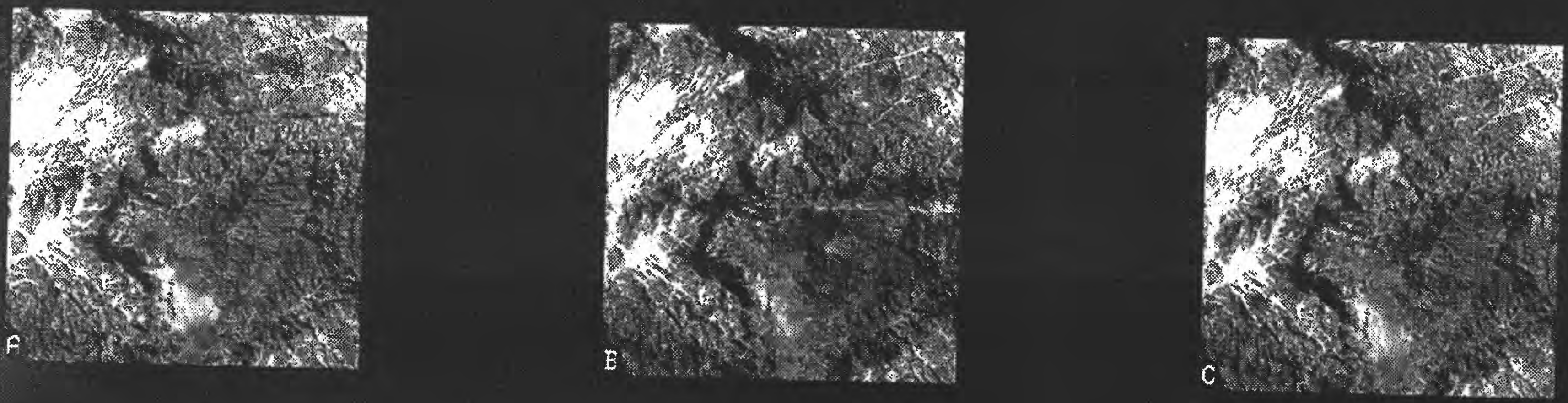


Fig. 22 - A TIMS image of the De Weese Plateau, Colorado (22A), is cut in half (horizontally) and a 'u-shaped' filter is applied to each half. When the half images are appended together, a horizontal 'join' effect is evident. Applying the same filter to overlapping segments and deleting the overlap areas during reassembly makes the effect much less evident.

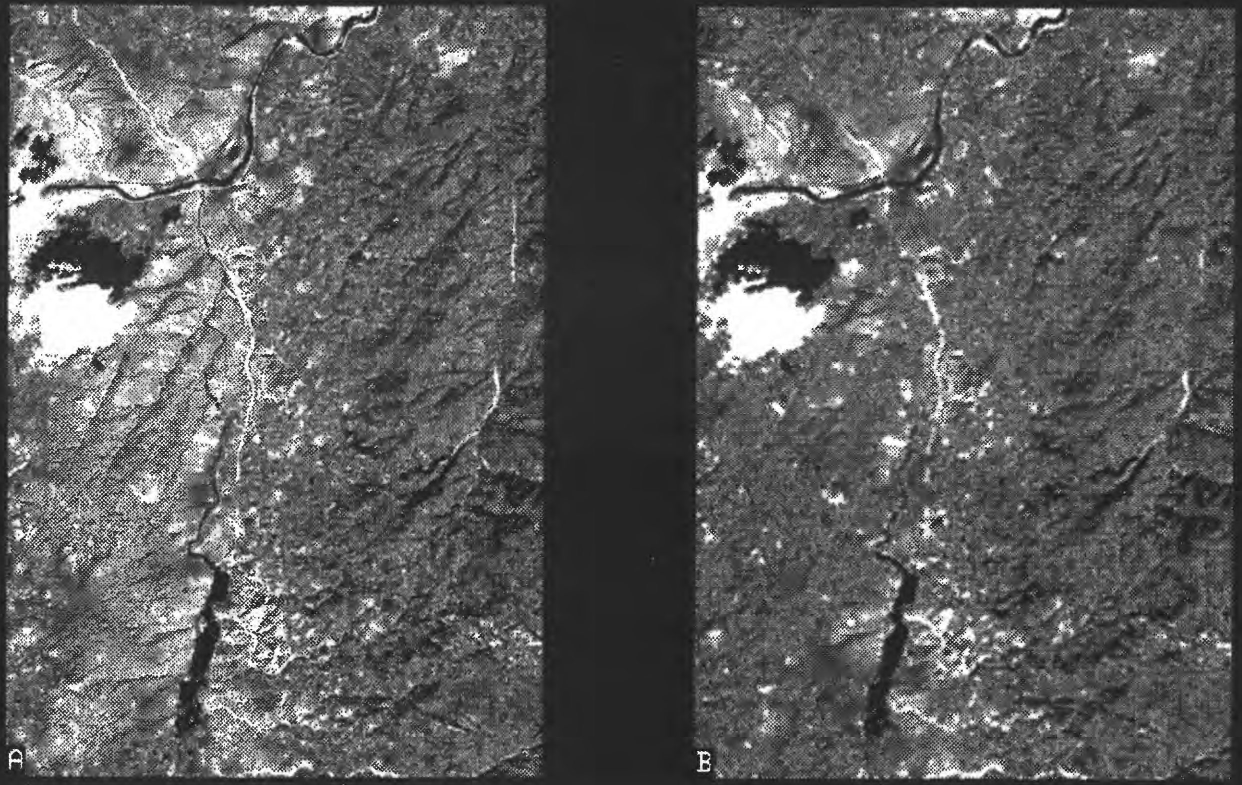


Fig. 23 - A SPOT image (A) of an area south of the town of Texas Creek, Freemont County, Colorado, and an accompanying low pass filtered version (B) which appears fuzzy.

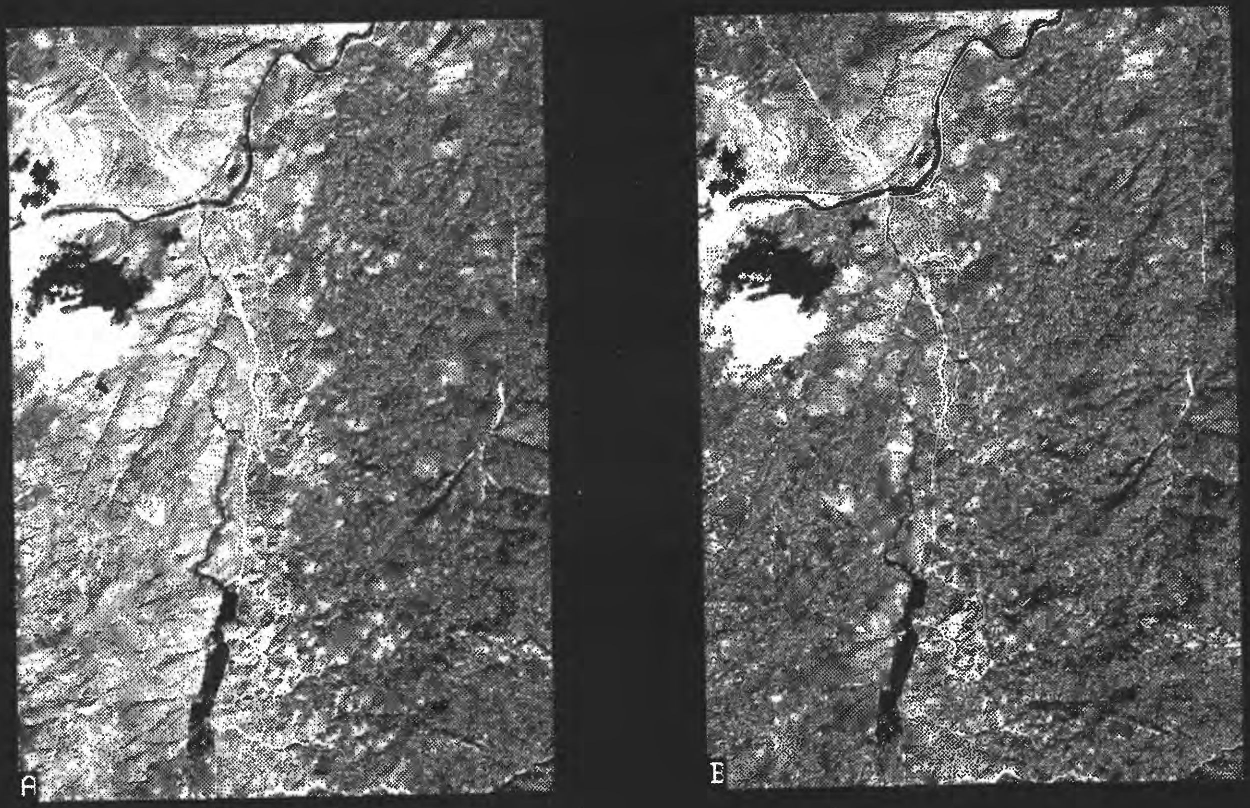


Fig 24 - The same example as Fig 23 (A) and an accompanying high pass enhancement filtered version (B). Some high frequency "ringing" artifacts can be observed.

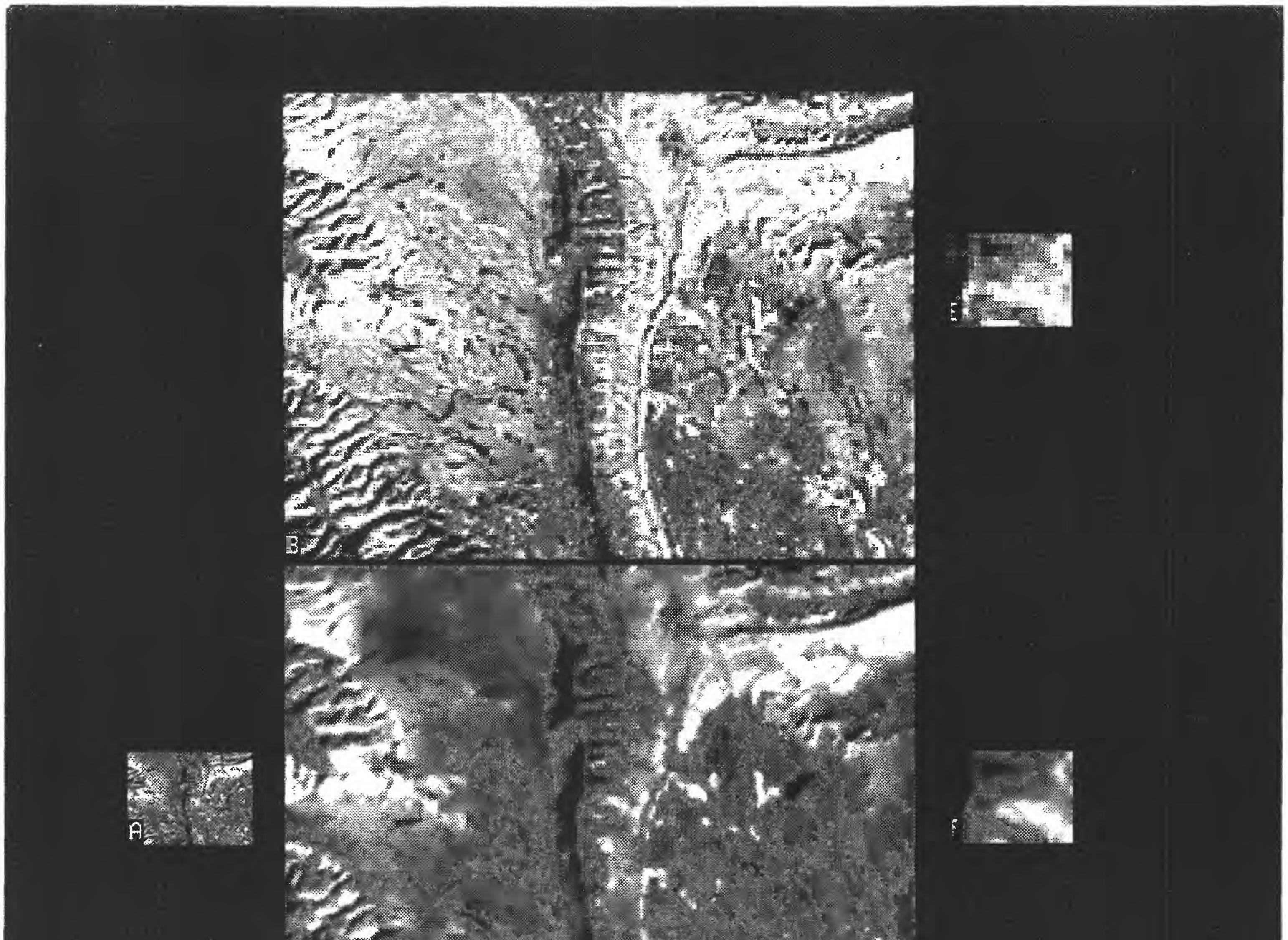


Fig. 25 - A TM image of the Canon City, Colorado, area (A) is enlarged first by a factor 5 (B,C,D) and then the center portions of these images by an additonal factor 5 (E,F,G) using three methods. The top panels (B,E) are enlarged using pixel replication. The middle panels (C,F) are expanded using bilinear interpolation. The lower panels (D,G) are enlarged using the FFT method.

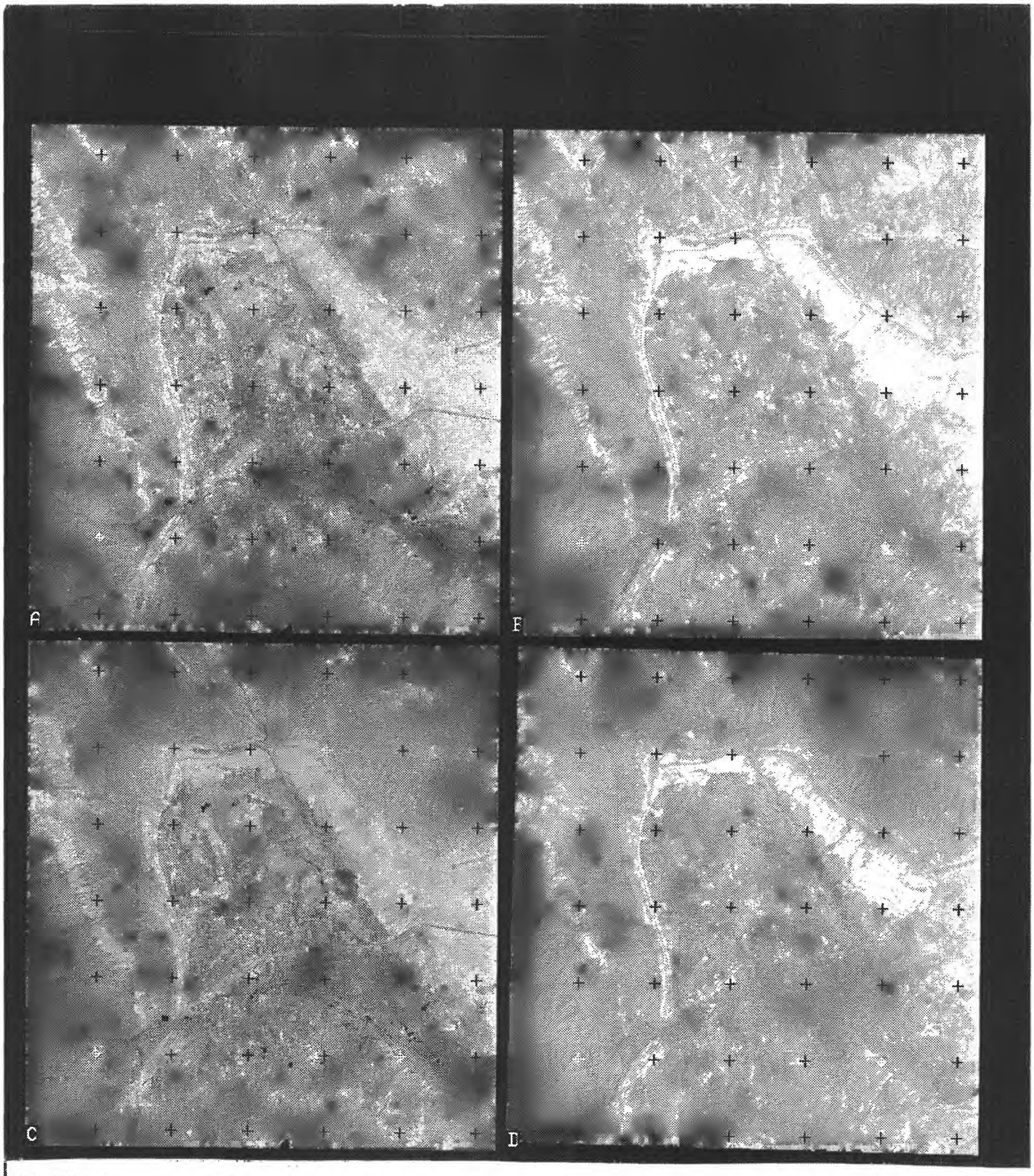


Fig. 26 - A cross-correlation registration of TM to SPOT data of the Canon City, Colorado, area using the FFT. To show the registration more clearly, the image dynamic range was compressed and a grid of black fiducial marks superposed. A and C are identical views of the SPOT image. B and D are the unregistered and registered TM images. Three circled fiducial marks, coded in white, illustrate the registration changes.

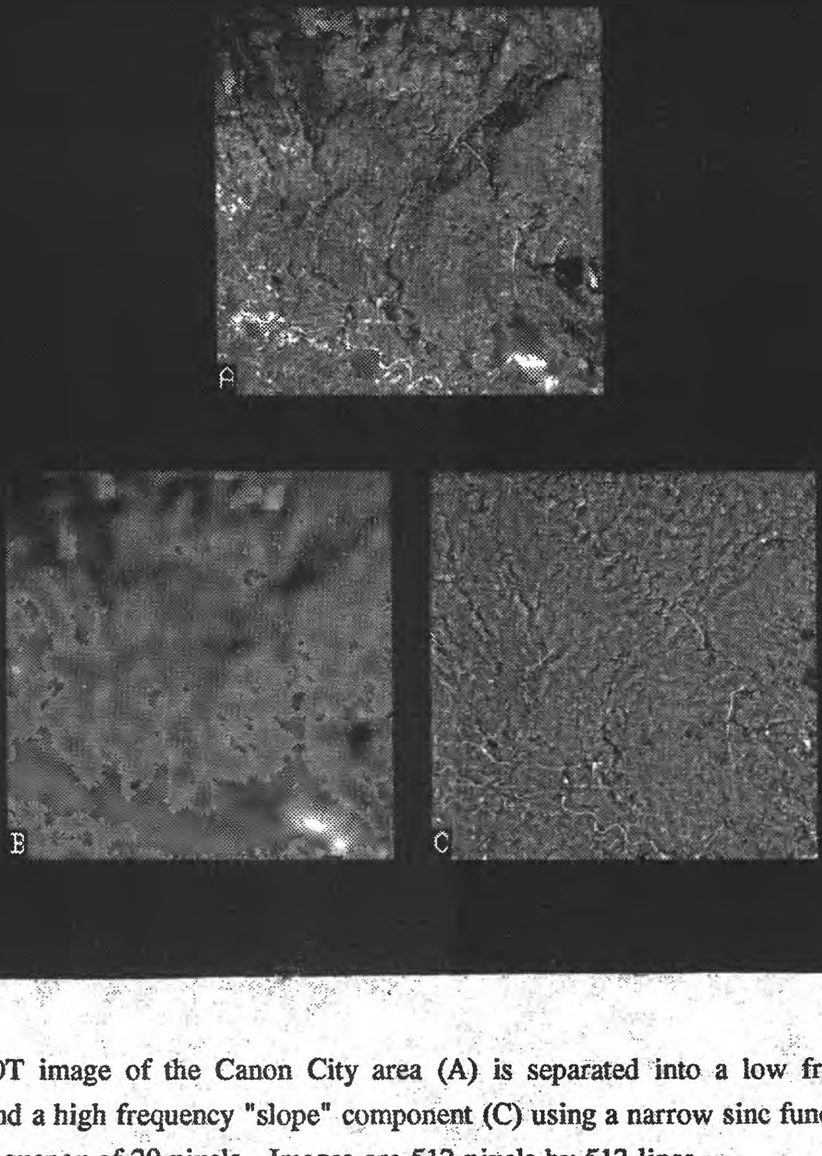


Fig. 27 - A SPOT image of the Canon City area (A) is separated into a low frequency "albedo" component (B), and a high frequency "slope" component (C) using a narrow sinc function positioned at a radial spatial frequency of 20 pixels. Images are 512 pixels by 512 lines.