NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMPLEMENTATION OF THREE SPECKLE REDUCTION FILTERS FOR SOLID PROPELLANT COMBUSTION HOLOGRAMS

by

Thomas D. Edwards

December 1986

Thesis Advisor: J. P. Powers

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Rocket Propulsion Lab; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Edwards, CA 93523	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
F04611-86-X-0008			

11 TITLE (Include Security Classification) IMPLEMENTATION OF THREE SPECKLE REDUCTION FILTERS FOR SOLID PROPELLANT COMBUSTION HOLOGRAMS

12 PERSONAL AUTHOR(S) Edwards, Thomas D.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1986 December	57

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Speckle; Speckle Index; Hologram; Laser; Image Processing; ITEX/PC

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Speckle noise appears when a diffuse light hologram of a particle field is reconstructed with a laser. The speckle must be reduced so that particles of interest may be counted and sized. This thesis describes speckle and a figure of merit known as the speckle index. Three speckle reduction filters suggested by work in the area of synthetic aperture radars were implemented, discussed and compared. An IBM AT personal computer with Itex/PC image processing software was used for the comparison. The geometric filter performed best overall, but may not be superior in every circumstance.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof J. P. Powers	22b TELEPHONE (Include Area Code) (408)646-2679	22c OFFICE SYMBOL 62Po

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

Implementation of Three Speckle Reduction Filters for Solid Propellant Combustion Holograms

by

Thomas D. Edwards
Captain, United States Marine Corps
B.S., U.S. Naval Academy, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1986

ABSTRACT

Speckle noise appears when a diffuse light hologram of a particle field is reconstructed with a laser. The speckle must be reduced so that particles of interest may be counted and sized. This thesis describes speckle and a figure of merit known as the speckle index. Three speckle reduction filters suggested by work in the area of synthetic aperture radars were implemented, discussed and compared. An IBM AT personal computer with Itex/PC image processing software was used for the comparison. The geometric filter performed best overall, but may not be superior in every circumstance.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

This thesis is a continuation of research being done at the Naval Postgraduate School. An investigation is underway to study the addition of aluminum and other metallic particles to solid rocket propellants in the hope of achieving greater performance characteristics. Performance is sensitive to the particle size distribution throughout the rocket motor and nozzle. Since no data exists to predict the distribution in this region of the motor, direct observation has been undertaken to provide the needed information.

Holographic techniques have been employed to capture the dynamics of the combustion chambers of small rocket motors while firing. These techniques are being refined and upgraded. Concurrently, improvements in the processing of the holograms to extract the particle size distributions are also necessary.

Previous work done in processing the images has included the use of a Quantimet 720 [Ref. 1], and more recently, the use of an IBM PC/AT with special software and hardware [Ref. 2]. The latter method has proven to be the best currently available. In this method, holograms are taken of the rocket combustion chamber while the propellant is burning. The holograms are then reconstructed. The reconstructed images are stored digitally and later processed to extract the needed particle size data.

A problem arises during the processing of the digital image, however. During laser reconstruction of the hologram, speckle noise is introduced. This speckle noise can be of the same size as some of the smaller particles of interest, and so cause a faulty count in the number of features present.

In an attempt to reduce the speckle, Reference 2 investigated a spinning mylar disk, averaging techniques, blurring and lowpass filtering. Twenty microns was the best resolution that was achieved by these means. Some of the loss in resolution may have been due to the optics involved; nevertheless, it became necessary to explore other means of filtering out the speckle to get the best possible resolution.

This thesis explores three speckle reduction algorithms suggested by work in the field of synthetic aperture radars. The initial step was to write computer programs to support each algorithm. Once the programs were running, a comparison was done to find which of the three was best at reducing speckle while retaining as much resolution as possible.

This thesis is divided into six chapters. Chapter Two describes the process involved in reconstructing a hologram, describes the Itex/PC driver software routines available to embed into user programs, and provides other background material. Chapter Three gives a description of speckle and introduces the "speckle index", a figure of merit in determining how much speckle is present. The description of each of the speckle reduction filters is given in detail in Chapter Four. Chapter Five is where the filters are compared with each other. They are ranked on speed, speckle reduction based on speckle index, and performance based on histogram and visual means. The concluding remarks are contained within Chapter Six.

II. HOLOGRAM IMAGE PROCESSING

An IBM PC/AT is the heart of the entire process of image manipulation, from digitizing an image to speckle reduction to the counting of the features. Installed on the IBM PC/AT are a PC Vision frame-grabber board, ImageAction and Itex/PC software (all by Imaging Technology, Inc.) the computer monitor, a video monitor and a video cassette recorder.

A video image from the VCR can be "grabbed" by the frame-grabber under control of the special software. Each of the picture elements (pixels) in the 512x480 array comprising a video image is assigned an integer gray level from 0 to 255 by the frame-grabber. Level 0 on the gray scale is blackest-black. Level 255 is whitest-white, while values in between are various shades of gray. Each pixel is uniquely addressable and its gray level alterable, thereby allowing the capacity to achieve digital filtering.

A. SOFTWARE

1. ImageAction

The ImageAction software is a set of menu-driven routines for use with a mouse. A totally closed system, it can perform image graphics, image analysis, image processing and filtering. Particularly useful are the "grab" routine described in the preceding paragraph, and the "histogram" routine which outputs the image statistics (standard deviation, mean, variance) along with the image's histogram. More details are described in Reference 2.

2. Itex/PC

The Itex/PC software performs most of the same functions as the ImageAction software.[Ref. 3] An important advantage is that the Itex/PC Pascal subroutines are able to be called up in Microsoft versions of Pascal, C or Fortran programs. This allows the user a great deal of flexibility in programming.

Itex/PC consists of subroutines that may be called from a main program to perform a specific function or to return a certain value. For example, if the subroutine

```
CALL THRESH (lowcut,highcut)
```

is used to threshold an image, all the pixel values between the integers "lowcut" and "highcut" will be displayed as a new value of 255 (white). All other pixels will be displayed as 0 (black). If, as in this thesis, "highcut" is set to 255 and "lowcut" varied by the user during run time, a binary image results which represents the black particles on a white background.

Sometimes an Itex/PC routine is treated as a function. An example is

```
L = RPIXEL (x,y).
```

Here, the gray level of the pixel located at (x,y) is read and the value is assigned to the integer L. The value of L may then be varied by one of the filtering algorithms and the new value written back into the pixel at (x,y) using

```
CALL WPIXEL (x,y,L).
```

Some specifics of Itex/PC bear mentioning. A standard television screen is made up of a 512x512 array of pixels. Of these, only 512x480 are actually visible. The lowest 32 rows are hidden out of view and may contain other information. Itex/PC labels the upper left-hand pixel as (0,0), rather than (1,1) as in most conventional arrays. The lower right-hand corner of visible pixels is therefore (511,479). This method of labeling can cause some confusion, and requires more attention to detail when writing programs which need to address specific pixels.

On the monochrome PC Vision frame-grabber board, four look-up tables (LUT's) may be utilized. A LUT serves to transform pixel values before they are displayed. The standard LUT is linear, meaning that the output values equal the input values. Certain actions, such as thresholding, change the LUT values. The actual values of the pixels in the frame-grabber board memory remain unchanged by the threshold, but are simply altered by the LUT in such a way that a binary image appears on the screen. If it is desired to make the threshold permanent and actually change the frame-grabber memory to their new values, the MAPLUT subroutine must be called. If not, a call to INITIA to initialize LUT's returns the screen to its "before-threshold" likeness.

3. Performance Comparison

Reference 2 describes a procedure to extract the particle size distribution from a reconstructed holographic image. The individual particles were numbered, sized in the x and y directions, and sorted into bins depending on their size. Non-round particles were included or excluded, at the users option. Due to limitations in the software available at that time, only one-quarter of the video screen could be processed in a timely manner. A full screen image was "quad-squished", thereby losing every other row and every other column of pixels. Obviously, resolution was degraded by a considerable amount.

When Itex/PC and MATLAB (matrix manipulation software) became available, the computer program of Reference 2 was upgraded. Prior to Itex/PC, it took approximately 40 minutes to process a "quad-squished" image. After Itex/PC, a full screen image was processed in approximately eight minutes. Taking into account the "quad-squish", this represents a twenty-fold decrease in processing time.

B. HOLOGRAPHY

As mentioned in the introduction, holographic techniques are used to capture the image of a rocket combustion chamber during firing. The pulsed ruby laser utilized in the recording process uses a glass diffuser in its illumination path. This diffuser is necessary to cut down the presence of schlieren interference fringes produced by the thermal and density gradients surrounding the burning particles in the rocket motor. [Refs. 4 and 5]

The reconstruction process is shown in Figure 2.1. The hologram is reconstructed using a krypton laser and viewed through the microscope using 2x, 4x or 10x magnification. The diffuse light from the recording interferes with the reference wave of the krypton reconstruction laser. This random interference causes speckle to be introduced into the image. By attaching a 0.5 lux camera and a video cassette recorder to the microscope, the speckle-corrupted image is recorded and preserved for later use. More detailed descriptions of the reconstruction process are in References 4 and 5.

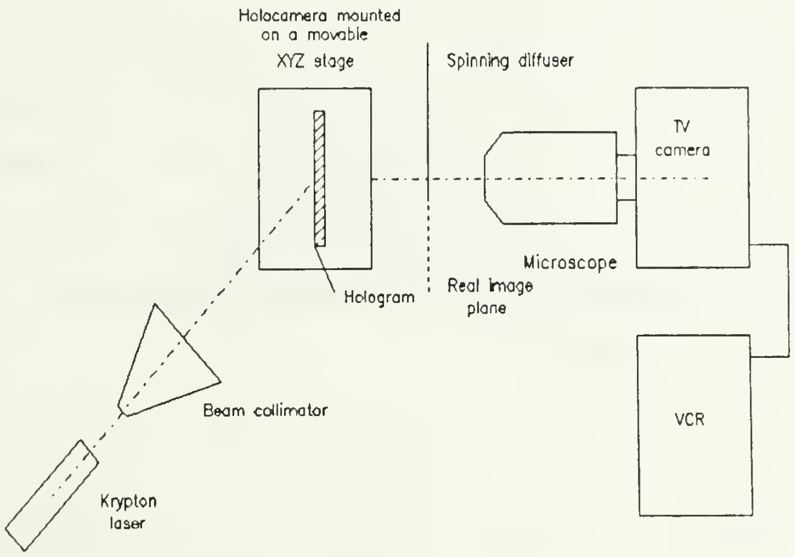

Figure 2.1 Hologram Reconstruction [Ref. 4]

Once on VCR tape, the desired image is "grabbed" by either ImageAction or Itex/PC software as described earlier. The digitized image then may be filtered immediately or stored on floppy disk (SAVEFT routine) for later use. The next step is to filter out as much speckle as possible without reducing resolution.

C. HISTOGRAMS

A histogram of an image consists of a bar graph. A separate bar exists at each gray level, with the height of the bar proportional to the number of pixels at that gray level. Thus, a histogram gives a visual representation of the general darkness or lightness of the image and how widely separated in gray level certain features may be. It can suggest where the best level to place a threshold might be.

D. THRESHOLDING

Ideally, the feature data in an image would be very different from the background in gray level, as in Figure 2.2. A threshold could then be placed at a level midway between them with the result that the feature particles would be black and the background white. The program developed in Reference 2 could then be used to properly size and count the particles of interest.

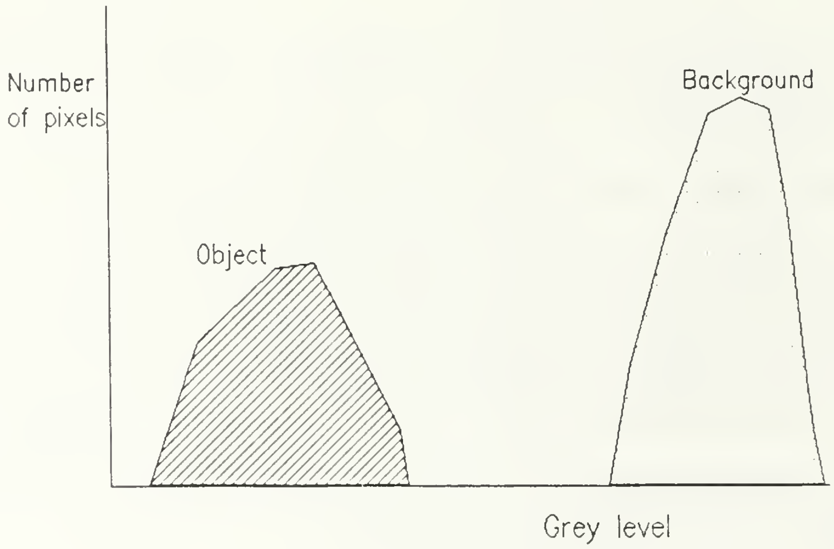

Figure 2.2 Ideal Histogram

In practice, Figure 2.3 is much more likely. Particles cover much of the range of gray levels, as does the background speckle. A threshold placed in an unfiltered image would cause portions of some particles to disappear while part of the speckle would appear as particles.

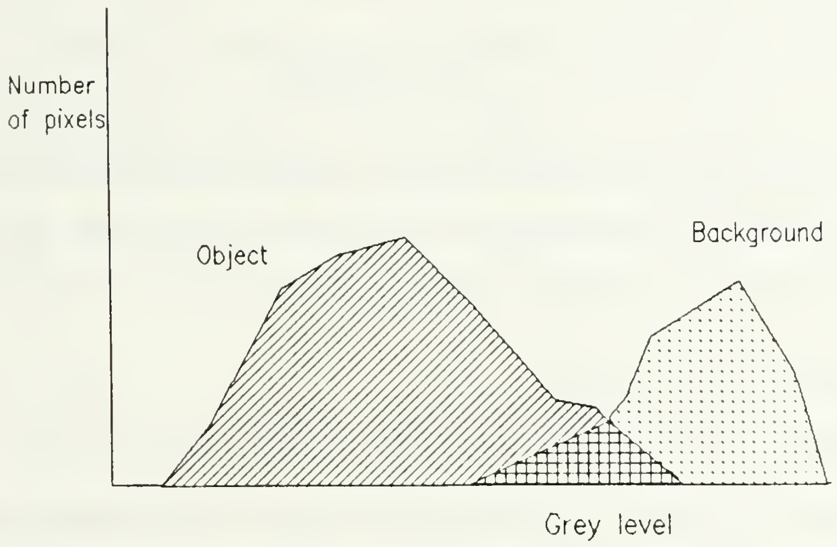

Figure 2.3   Histogram Showing an Overlap of Gray Levels

What is required is a method to process out the unwanted speckle noise in the background. With a clear separation between the particles of interest and the background, a threshold could then be applied with success.

III. SPECKLE

The purpose of this thesis was to implement filters for the reduction of speckle. Therefore, it was natural to attempt to quantify the amount of speckle reduction with each iteration of a given filter.

A. DESCRIPTION OF SPECKLE

Speckle has the characteristics of random multiplicative noise in the sense that the noise level increases with the average gray level of a local area [Ref. 6]. The presence of speckle reduces ones ability to resolve fine detail.

The following description may help to give an intuitive idea of what speckle is. [Ref. 6] First, imagine a two-dimensional image, such as a television screen, laying horizontally on a flat surface. The gray level of each individual pixel is represented by a vertical tower, thus making a three-dimensional image. The height of each tower is proportional to its gray level; darker colors (lower numbers) being short towers and lighter colors (higher numbers) being tall towers. Since there are 512x480 (245,760) individual towers, the impression from a distance is of a landscape with smoothly flowing valleys and hills.

In this imagined landscape, a dark feature would appear as a wide, deep valley. The darker the feature, the deeper the valley would be. Similarly, a light feature would rise from its surroundings like a skyscraper.

Now, short, narrow, winding gullies and ridges are added to the landscape. Wormy and random in appearance, they corrupt the smoothness of the surroundings. This is speckle noise.

Figure 3.1 shows an image with and without speckle present. It can be seen how the speckle is easily confused with the smaller feature particles.

B. SPECKLE INDEX

As a figure of merit in determining the amount of speckle reduction, the "speckle index" was used. In Reference 7, Crimmins showed that the ratio of local deviation to local mean was a reasonable measure, due to the multiplicative nature of speckle noise.

1. Algorithm

To compute the speckle index, a fortran subroutine was written. The algorithm used was: [Ref. 7]

$$\text{speckle index} = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \frac{\text{dev}}{\text{mean}} \tag{3.1}$$

where:

$$\text{mean} = \frac{1}{9} \sum_{a,b=-1}^{1} p(x+a, y+b) \tag{3.2}$$

$$\text{dev} = \max_{-1<a,b<1} p(x+a, y+b) - \min_{-1<a,b<1} p(x+a, y+b) \tag{3.3}$$

In the above equations, M and N are the dimensions of the array in the x and y directions, and p(x,y) is the gray level of an individual pixel located at x,y.

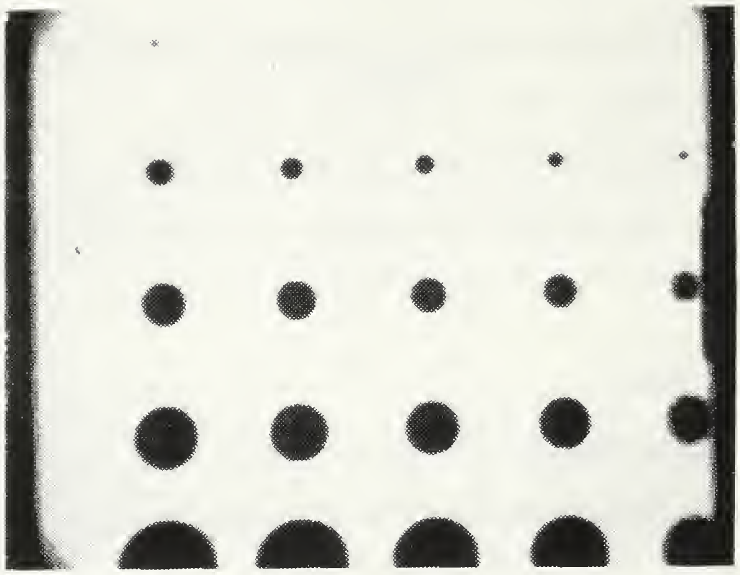

Figure 3.1a   Sample Image without Speckle

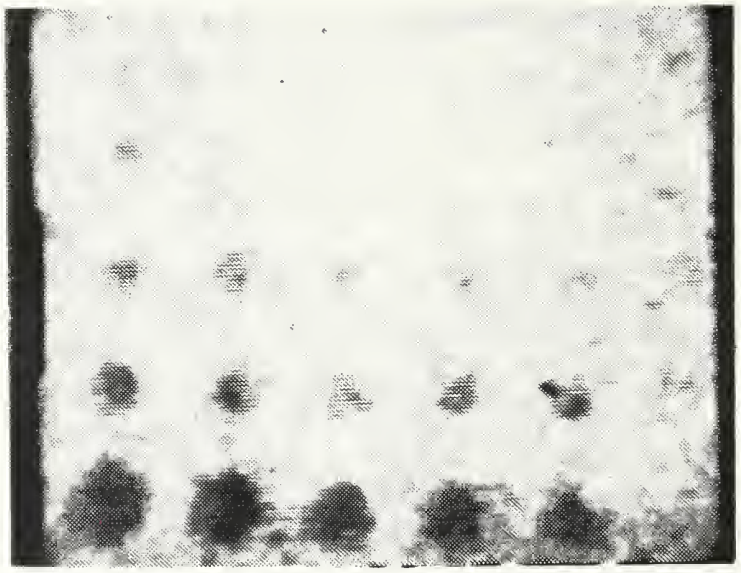

Figure 3.1b   Same Image with Speckle Noise Added

In simple terms, the local deviation is found by subtracting the smallest gray level of a particular pixel and its eight immediate neighbors from the largest gray level of the same nine pixels. The local mean is the average of the nine gray levels.

2. <u>Implementation</u>

The complete fortran subroutine to calculate the speckle index of an image is included in Appendix A. The subroutine was run with various values of M and N to determine a good array size. From Figure 3.2, it can be seen that a small array can distort the calculation if there is a sharp local disturbance. The values M = N = 240 were chosen as representative array dimensions for speckle index calculations. This size represents about 23% of the visible screen, yet does not take an excessive amount of time to calculate.

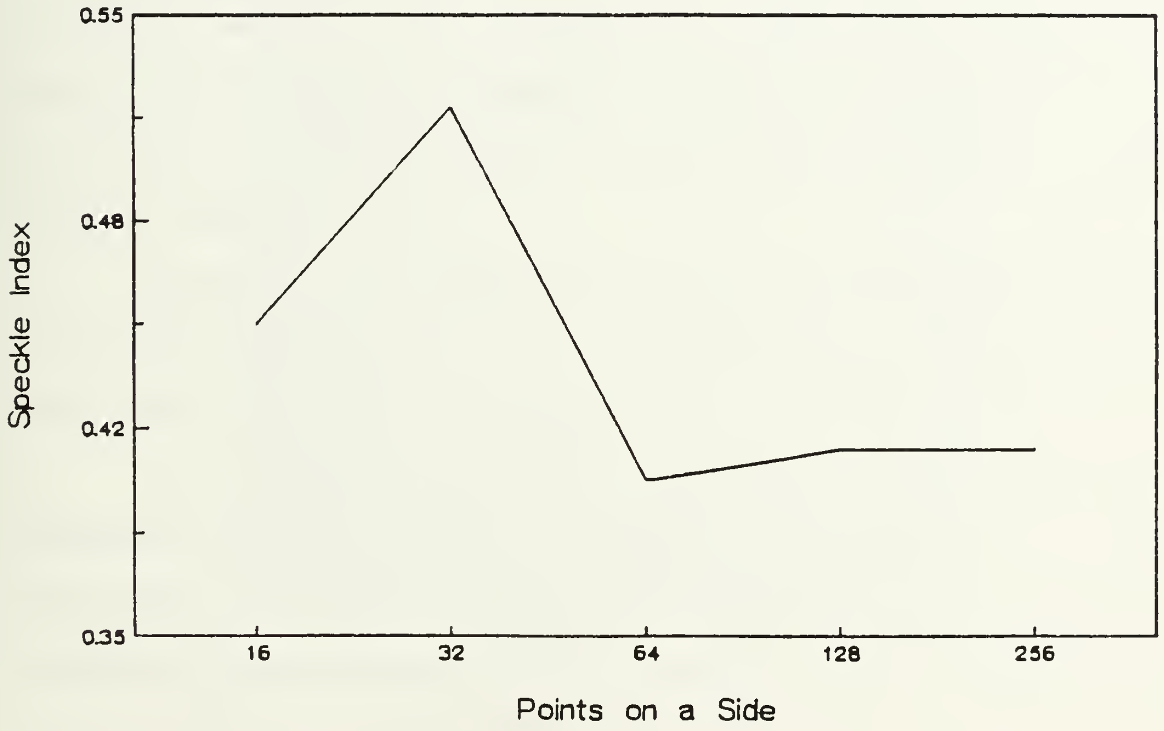

Figure 3.2 Speckle Index as a Function of Sample Size

3. Performance

The IBM PC/AT with the math coprocessor returns the speckle index to the calling program in 75 seconds.

Values for speckle index range from over 1.0 (rarely) to a theoretical limit of zero. The value of zero could only be achieved if the entire image was filtered to the point where every pixel had the same gray level. This is, of course, uninteresting.

IV. THE SPECKLE REDUCTION FILTERS

Three filters were implemented in software and will now be discussed. All of them call subroutines "speckle" [Appendix A] and "trash" [Appendix B] to calculate the speckle index and threshold the image, respectively.

Two filters depend to a certain degree on the statistics of the image. The third filter uses a geometric hulling algorithm.

A. THE SIGMA FILTER

1. Description

The first speckle reduction filter is based on the standard deviation of a Gaussian distribution. By definition, 95.5% of all the pixels fall within 2 standard deviations on either side of the mean. Any pixels within 2 standard deviations of a given pixel's gray level are included in an averaging scheme, whereas those outside the "2-sigma" range are excluded. [Ref. 7]

Obviously, the standard deviation of the image must be known beforehand. ImageAction was used to find standard deviations and histogram plots very quickly and easily.

If a particular pixel is considerably different from its neighbors, perhaps none of the neighbors will be within the 2-sigma range. This would indicate a very sharp feature. To avoid the possibility that this sharp feature will not be subject to the averaging process at all, a cutoff is established. If the total number of pixels inside the 2-sigma range is less than this minimum cutoff number (2 was chosen), the four-neighbor average then replaces the central pixel's gray level value.

2. Algorithm

The sigma filter program is included [Appendix C]. It uses the algorithm as follows: [Ref. 8]

$$g(x,y) = \begin{cases} \dfrac{\sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} \delta(i,j)p(i,j)}{\sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} \delta(i,j)} & \text{if } \sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} \delta(i,j) > 2 \\ \dfrac{p(x-1,y)+p(x+1,y)+p(x,y)+p(x,y-1)+p(x,y+1)}{5} & \text{otherwise} \end{cases} \tag{4.1}$$

where: p(x,y) = gray level of a pixel in 5x5 local array

g(x,y) = gray level of the filtered central pixel

$$\delta(x,y) = \begin{cases} 1 & \text{if } (1-2\sigma)p(x,y) \leq p(i,j) \leq (1+2\sigma)p(x,y) \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

σ = standard deviation

3. Discussion

By altering the value of σ , varying degrees of filtering result. If σ is increased, the sigma range is increased, and so more pixels are included in the average. For this paper, however, the true standard deviation of each image was used when it was filtered.

For illustration of the sigma filter's effectiveness, images of the Air Force Resolution Target are presented. Figure 4.1 shows an image before and after two iterations of the filter. Visually the speckle has been reduced considerably, although the edges have been blurred and the resolution has decreased somewhat.

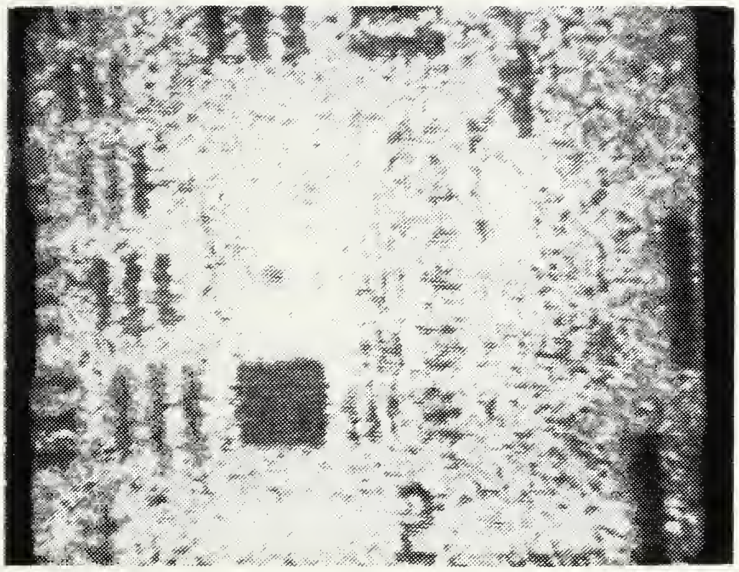

Figure4.1a Unfiltered Image of the Air Force Resolution Bar Target (S.I.= 0.30549)

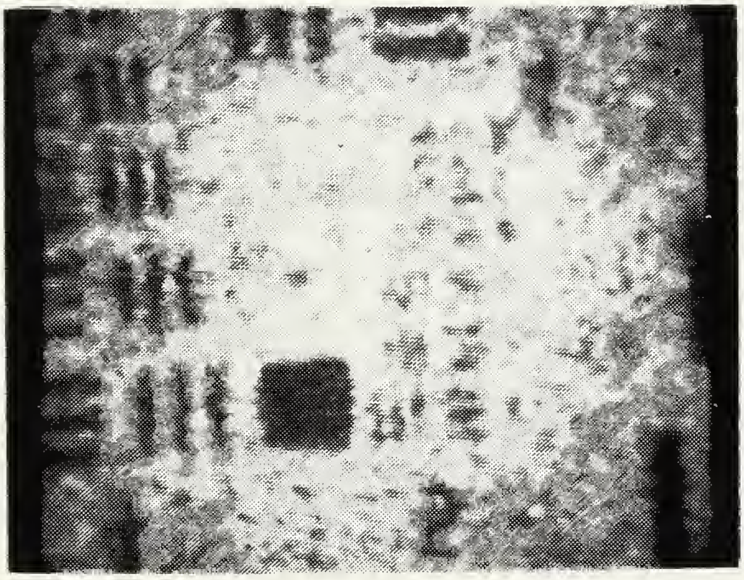

Figure 4.1b Same Image Filtered with the Sigma Filter (2 iterations, S.I.= 0.10278)

Histograms of these two images (Figure 4.2) show some separation between features and the background speckle after filtering. A threshold at 140 is suggested.

When this threshold is applied (Figure 4.3), the effectiveness of the sigma filter is most dramatically displayed. Now it is much more evident that good data can be retrieved from the filtered image, whereas the excessive speckle in the unfiltered image would lead to faulty data detection.

B. THE LOCAL STATISTICS FILTER

The program to implement this filter is included as Appendix D.

1. Description

This filter uses local estimates of the mean and variance in a 5x5 window about the central pixel in question [Ref. 8]. These local statistics are used to calculate a weight, k. The k value then determines where the new gray level of the central pixel will be placed: near the original value of the central pixel, near the linear average of all pixels in the 5x5 array, or somewhere in-between.

2. Algorithm

Throughout this algorithm, p(x,y) represents unfiltered pixels and g(x,y) represents the filtered central pixel. To compute the filtered gray level of a pixel using the local statistics method, the local mean, E{g(x,y)}, and variance, var{g(x,y)}, must first be estimated: [Ref. 8]

$$E\{g(x,y)\} \approx \frac{1}{25} \sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} p(i,j) \qquad (4.3)$$

Figure 4.2a Histogram of the Unfiltered Image

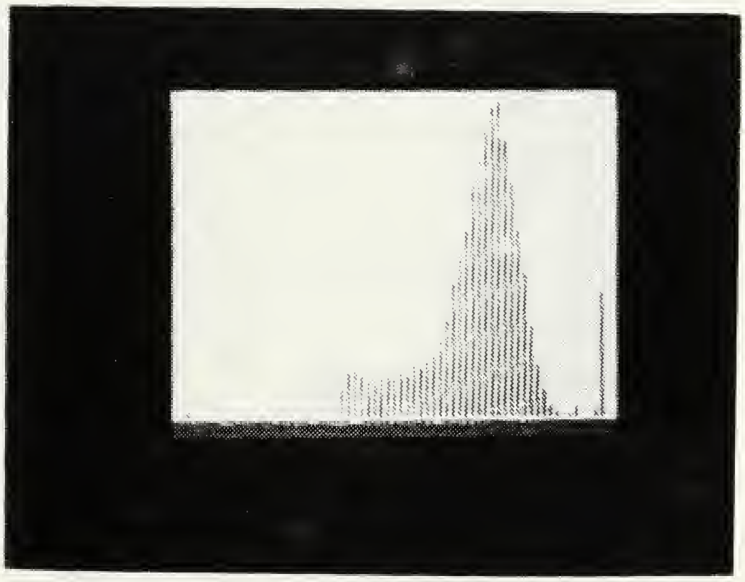
Figure 4.2b Histogram of the Sigma Filtered Image

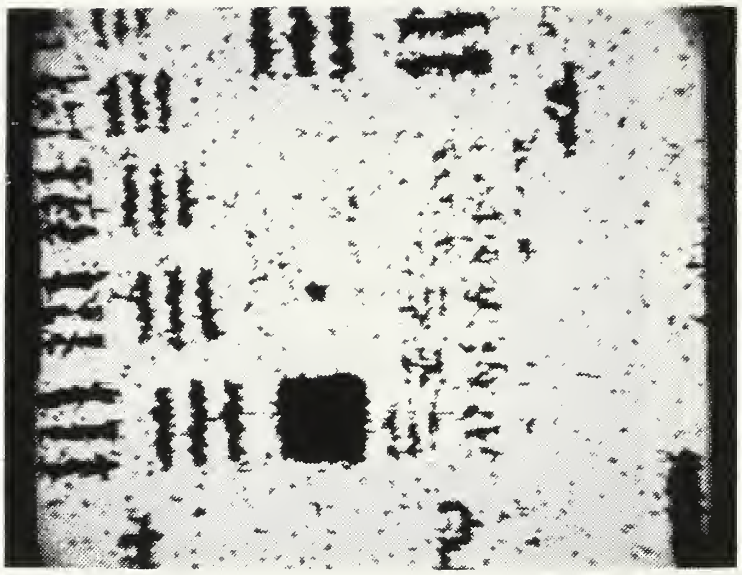

Figure 4.3a Unfiltered Image Thresholded at 140

Figure 4.3b Sigma Filtered Image Thresholded at 140

$$\mathrm{var}\{g(x,y)\} \approx \frac{\mathrm{var}\{p(x,y)\} + E^2\{g(x,y)\}}{\sigma^2 + 1} - E^2\{g(x,y)\} \tag{4.4}$$

where:

$$\mathrm{var}\{p(x,y)\} \approx \frac{1}{25} \sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} [p(i,j) - E\{g(x,y)\}]^2 \tag{4.5}$$

The weight k may then be calculated as:

$$k = \frac{\mathrm{var}\{g(x,y)\}}{\sigma^2 E^2\{g(x,y)\} + \mathrm{var}\{g(x,y)\}} \tag{4.6}$$

Finally, the estimate of g(x,y) is:

$$g(x,y) = E\{g(x,y)\} + k\,[p(x,y) - E\{g(x,y)\}] \tag{4.7}$$

From close scrutiny of the algorithm, some of the equations may be better understood. Equation 4.3 states that the local mean is the average of the 25 pixels in the 5x5 local array. Equation 4.5 is the variance of the unfiltered pixels in the 5x5 array; the mean value of the local array from Equation 4.3 is subtracted from each of the pixels in turn. These differences are squared and then summed over the entire array. For more detail on how the remaining equations were developed, see Reference 8.

3. Discussion

Consider a region of an image which is flat, meaning adjacent pixels have about the same gray level. Here, the variance approaches zero, and so k approaches zero. The new estimate of g(x,y) is therefore close to the mean of the local 5x5 array. Consider now a region of high contrast, say the edge of a particle. Here, k approaches one and so g(x,y) is close to the value of the central pixel. The filter therefore has a marked tendency toward the retention of high contrast edges. [Ref. 8]

Figure 4.4 shows the Air Force Resolution Target after two iterations of the local statistics filter, both before and after thresholding at 140.

C. THE GEOMETRIC FILTER

The program to implement this filter is included as Appendix E.

1. Description

Whereas the previously discussed filters depend to a certain degree upon the statistics of the image to be filtered, the geometric filter is based on non-linear geometric concepts.

The algorithm was developed in References 6 and 7. It is a one-dimensional routine which is run horizontally, vertically and then in the two diagonal directions. It applies in each direction a geometric hulling algorithm to the image and then to the image's complement.

2. Algorithm :[Ref. 6]

1) Let a=1, b=0 (this sets the values for a horizontal run).

2) Let c=3 (this is a counter to determine when to change directions).

3) Let d=1 (this controls whether the image or its complement is being filtered).

4) $$g(x,y) = \max\left[p(x,y), \min\{p(x-a,y-b)-1, p(x,y)+1\}\right] \quad \text{for } 1 \le x \le M, 1 \le y \le N \tag{4.8}$$

5)

$$p(x,y) = \max\left[g(x,y), \min\{g(x-a,y-b), g(x,y)+1, g(x+a,y+b)+1\}\right] \quad \text{for } 1 \le x \le M, 1 \le y \le N \tag{4.9}$$

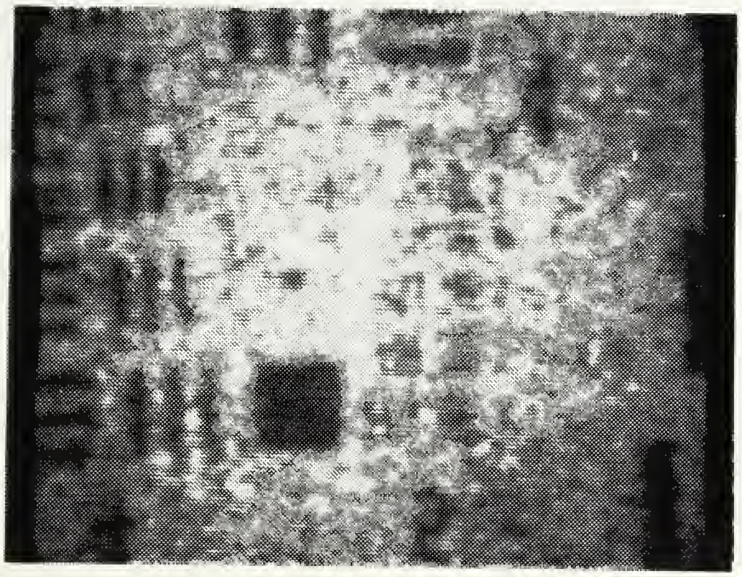

Figure 4.4a   Image Filtered with the Local Statistics Filter
(2 iterations, S.I.= 0.11484)

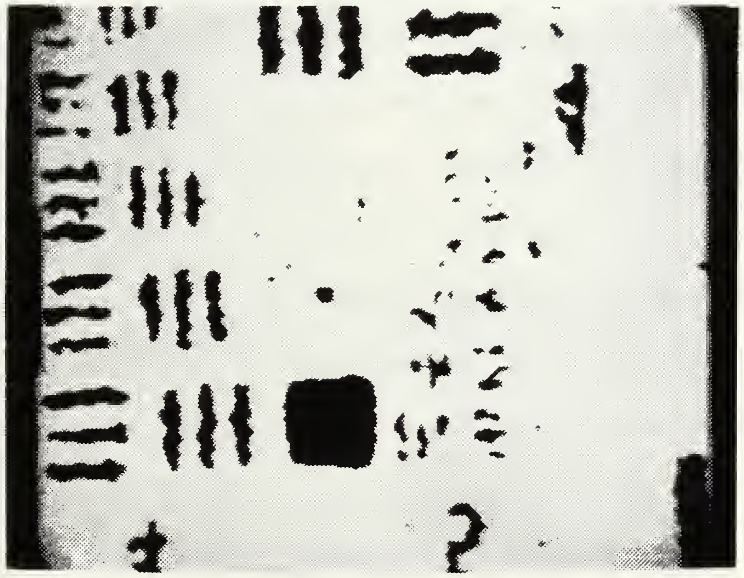

Figure 4.4b   The Local Statistics Filtered Image
Thresholded at 140

6) If d=1, let a=-a, b=-b, d=0. Go to step 4).
If d=0, let d=1. Go to step 7).

7)
$$g(x,y) = \min\left[p(x,y), \max\{p(x-a,y-b)+1, p(x,y)-1\}\right] \quad \text{for } 1 \le x \le M, 1 \le y \le N \tag{4.10}$$

8)
$$p(x,y) = \min\left[g(x,y), \max\{g(x-a,y-b), g(x,y)-1, g(x+a,y+b)-1\}\right] \quad \text{for } 1 \le x \le M, 1 \le y \le N \tag{4.11}$$

9) If d=1, let a=-a, b=-b, d=0. Go to step 7).
If d=0, let d=1. Go to step 10)

10) If c=3, let a=0, b=1, c=2. Go to step 4).(this sets up a vertical run)
If c=2, let a=1, b=1, c=1. Go to step 4). (this sets up a diagonal run)
If c=1, let a=1, b=-1,c=0. Go to step 4). (this sets up the other diagonal)

11) If c=0, stop.

3. Discussion

The algorithm used exhibits the interesting property that it will allow a certain curvature in the terrain (referring again to the "landscape" mentioned earlier), but will tear down anything more drastic. With a few exceptions [Ref. 6], a curve sharper than 45 degrees at any vertex will be filtered. What this means is that the narrow valleys comprising speckle will be filled in and the walls torn down after sufficient iterations of the filter. Of course, the objects of interest that should be preserved are also filled in and torn down, but at a much slower rate. Generally, the larger and darker the feature, the more slowly it will be degraded.

If an object of interest is almost as narrow as the speckle, they both will be reduced at about the same rate. Hopefully, the object is much darker than the speckle, so the speckle will be beaten down after a number of iterations leaving the object of interest largely intact.

Figure 4.5a shows the Air Force Resolution Chart after three iterations of the geometric filter. Figure 4.5b shows the same filtered image thresholded at a gray level of 140.

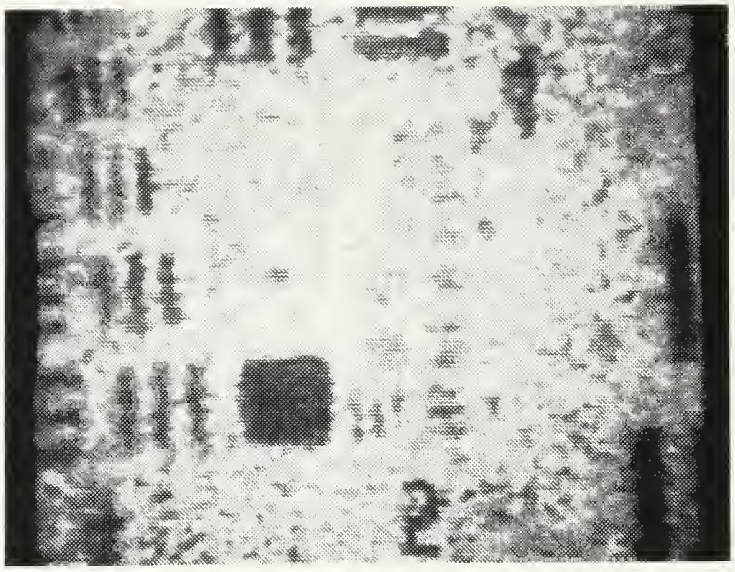

Figure 4.5a Image Filtered with the Geometric Filter
(3 Iterations, S.I.= 0.12918)

Figure 4.5b Geometric Filtered Image Thresholded at 140

V. COMPARISON OF THE FILTERS

This chapter will compare the three filters using the speckle index as a figure of merit in gauging how much speckle has been removed. The filters will be compared in how much speckle they remove per iteration, the time per iteration, histogram differences, and, of course, visual differences.

A. SPECKLE INDEX COMPARISON

All three filters reduce speckle at their own rate. Figure 5.1 shows this fact. The sigma and local statistics filters both reduce speckle by almost a factor of two after only one iteration, but then dramatically "slow down" with increased iterations. This may be attributed to the fact that both filters are actually various themes on the blurring technique. More iterations serve to merely increase the blur.

Figure 5.1 shows that the geometric filter's curve is more gradual, allowing the user to stop at a less severe level of filtering. This is a particularly desirable trait, because resolution degradation becomes a problem after only a few iterations. It should be noted that filtering cannot stop at an arbitrary value of speckle index, only at discrete levels depending on how many iterations have occurred. It is foreseeable that a given filter may allow a degree of speckle reduction not achievable by the others.

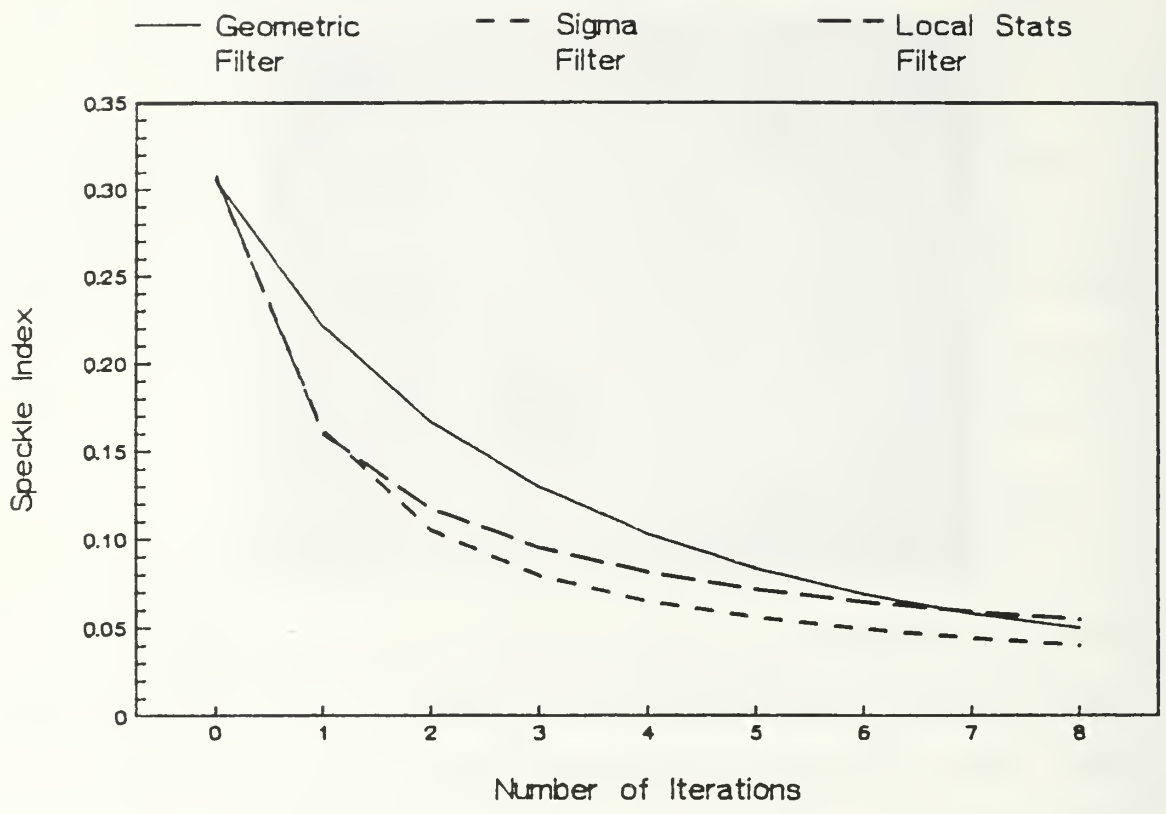

Figure 5.1 Speckle Reduction Comparison

On the basis of Figure 5.1, the geometric filter ranks first in its ability to control the amount of speckle reduction, followed by the local statistics filter.

B. TIME COMPARISON

The times for an iteration of each filter are in Table 4.1.

TABLE 4.1 TIMES PER ITERATION

FILTER	TIME
Sigma	7 min, 20 sec
Geometric	18 min, 50 sec
Local Statistics	39 min, 45 sec

The local statistics filter is slow due to the nested do-loops, frequent calls to subroutines and large number of calculations required inside each do-loop.

It should be noted that speed is of secondary importance since there is no requirement in this application to achieve results within a given time limit. Also, dedicated image-processing equipment can do the calculations much more efficiently and faster. For example, Reference 8 reports times of 14 seconds per iteration for the geometric filter using a VAX 11/780 and DeAnza IP-5500 digital video processor.

C. VISUAL COMPARISON

The ultimate test of any procedure dealing with images is how the image looks to the viewer. Comparisons of this type are difficult to justify, for each person sees an image slightly differently. Nevertheless, based on numerous runs of the filters, the geometric filter is judged best at retaining the edges, basic shape and size of objects of interest while beating down the speckle. This can be seen by close comparison of Figures 4.1b, 4.4a and 4.5a. All three Figures are close in speckle index, yet the local statistics and sigma filters tend to smear and spread particle edges more than does the geometric filter. This observation has been verified in runs using other images as well.

Once the threshold at 140 is applied (Figures 4.3b, 4.4b and 4.5b), the sigma filter is definitely seen as the inferior of the three. The other two are very close. Examination reveals a few small particles of speckle that the geometric filter failed to eliminate which the local statistics filter successfully handled. This is a consequence of the filtering stopping at discrete levels. In most other cases, the geometric filter is superior.

D. HISTOGRAM COMPARISON

All three filters produced histograms generally similar in shape and distribution (Figures 4.2b, 5.2a, 5.2b). Notable differences include the geometric filter's reduction of the spike at 255, the way it retained virtually intact the darker features and its more pronounced "valley" between the two "humps".

The claim was made earlier that the geometric filter had a tendency to reduce smaller (hopefully speckle) particles much faster than the dark features. The retention of the darker pixels in the histogram is evidence that this claim is valid.

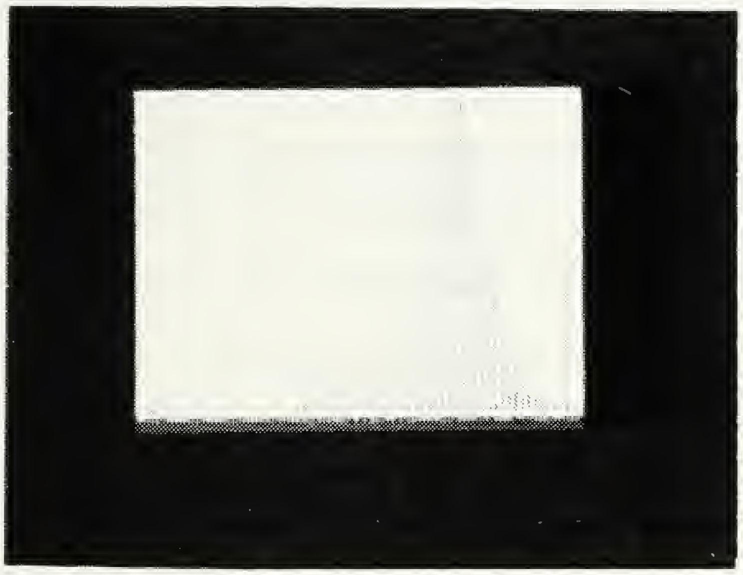

Figure 5.2a   Histogram of Local Statistics Filtered Image

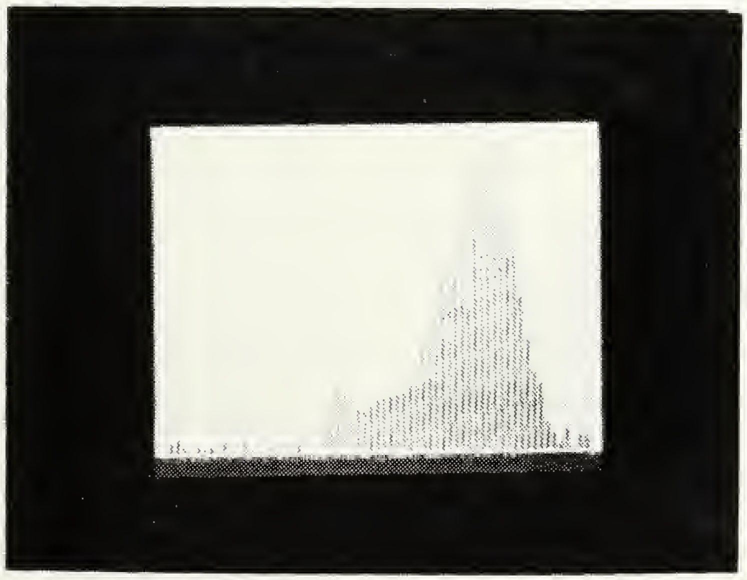

Figure 5.2b   Histogram of Geometric Filtered Image

All the filtered histograms pointed to a threshold at about 140. Thus, this was the value used on all the images, including the original, for comparison purposes.

VI. CONCLUSIONS

The IBM PC/AT with dedicated software and hardware is a viable system to reduce speckle in reconstructed holograms. The ITEX/PC software with the PC Vision frame-grabber board may be used in conjunction with fortran programs to achieve the filtering desired.

While all three filters are superior to current techniques, the geometric filter has been found to be the best of the three. It combines the ability to hold the edges and shape of objects with the desirable trait of less filtering per iteration. In this way, the user has more control over the amount of filtering to be done and to what degree the speckle will be reduced.

While the geometric filter has been found to be the best overall filter, it may not be superior in every particular circumstance. It has been noted that due to the discrete jumps in speckle index which occur after every iteration, one filter may be able to reach the region of optimum filtering when another cannot. For this reason, the local statistics filter should be used along with the geometric filter and the results after 2 or 3 iterations compared.

Resolution of the original image was about 12 microns. This was degraded to about 14 microns in the filtered images. It is doubtful that much more improvement can be made toward reducing the speckle degradation due to filtering. Any further resolution improvements most likely will have to occur in the recording and reconstruction process, and better optics. Some improvements have been made recently in this area.

Reference 4 reports resolutions typically of 8 microns, and in certain cases, 4 microns. Taking into account the degradation after filtering, a conservative estimate of 10 microns resolution is now possible in the finished filtered and thresholded image.

APPENDIX A.

SPECKLE INDEX SUBROUTINE

```
$include: 'itexpc.inc'
        subroutine speckle(spklindx)
c       ****************************************************************
c           This subroutine computes the speckle index (a figure of
c       merit) of the upper-left hand corner of the image, a 240x240
c       array.
c       ****************************************************************
c
c       *** DEFINITIONS ***
c       small       min gray value in 3x3 local array
c       big         max gray value in 3x3 local array
c       sum         summation of the nine gray values in 3x3 array
c       dev         difference between small and big
c       mean        average of the nine gray values
c       local       local deviation divided by the local mean
c       total       summation of local over entire input array
c       spklindx    speckle index
c
c       *** DECLARATIONS ***
c
        implicit integer*2 (a-z)
        integer*2 small,sum,big,dev
        real*4 mean,local,total,spklindx
c
  998   total=0.0
        do 40  m=1,238
           do 30  n=1,238
c
c       *** COMPUTE THE LOCAL MEAN AND DEVIATION ***
c
             big=max0(RPIXEL(m-1,n-1),RPIXEL(m-1,n),RPIXEL(m-1,n+1),
    %        RPIXEL(m,n-1),RPIXEL(m,n),RPIXEL(m,n+1),RPIXEL(m+1,n-1),
    %        RPIXEL(m+1,n),RPIXEL(m+1,n+1))
             small=min0(RPIXEL(m-1,n-1),RPIXEL(m-1,n),RPIXEL(m-1,n+1),
    %        RPIXEL(m,n-1),RPIXEL(m,n),RPIXEL(m,n+1),RPIXEL(m+1,n-1),
    %        RPIXEL(m+1,n),RPIXEL(m+1,n+1))
             dev=big-small
             sum=RPIXEL(m-1,n-1)+RPIXEL(m-1,n)+RPIXEL(m-1,n+1)+
    %        RPIXEL(m,n-1)+RPIXEL(m,n)+RPIXEL(m,n+1)+
    %        RPIXEL(m+1,n-1)+RPIXEL(m+1,n)+RPIXEL(m+1,n+1)
             mean=sum / 9.0
             if (mean .eq. 0) then
                  mean=1
             end if
             local=dev / mean
             total=total + local
   30      continue
```

```
40   continue
     spklindx=total/56644.
     write(*,*)'the speckle index is:',spklindx
     return
     end
```

APPENDIX B.

THRESHOLD SUBROUTINE

```
$include:'itexpc.inc'
        subroutine trash
c       ****************************************************************
c           This subroutine thresholds the current image to whatever
c       value the user decides. It will change this threshold value as
c       many times as is necessary.
c       ****************************************************************
c
c       *** DEFINITIONS ***
c       errval      error value for the file retrieval
c       limit       threshold level
c
c       *** DECLARATIONS ***
        implicit integer*2 (a-z)
        integer*2 errval,limit
c
c      *** SET BASE ADDRESS OF HARDWARE REGISTERS, INDICATE A PSEUDO-
c       COLOR BOARD AND INITIALIZE THE LUT's TO STANDARD VALUES. ***
        call SETHDW(16#ff00,16#c000,1)
        call INITIA
c
 20     write(*,*) 'Thresholding-- enter the desired pixel value'
        read(*,*) limit
c
c       *** THRESHOLD THE 4 LOOK UP TABLES (LUT'S) ***
        call SETLUT(0,0)
        call THRESH(limit,255)
        call SETLUT(1,0)
        call THRESH(limit,255)
        call SETLUT(2,0)
        call THRESH(limit,255)
        call SETLUT(3,0)
        call THRESH(limit,255)
c
c       *** QUERY USER ON WISHES TO THRESHOLD AGAIN ***
        write(*,*) 'do you wish to set a new threshold value? (0 for yes)'
        read(*,*) d
        if (d .eq. 0) then
                call INITIA
                go to 20
        end if
c       *** INITIALIZE THE IMAGE BEFORE RETURNING ***
        call INITIA
        return
        end
```

APPENDIX C.

TWO-SIGMA FILTER PROGRAM

```
$include:'itexpc.inc'
c
c       ****************************************************************
c           This is a statistical filter known as a "2-sigma". It calcu-
c       lates the range of gray levels within 2 standard deviations of
c       the central pixel in a 5x5 array. Any of the 24 pixels surround-
c       ing the central pixel that fall within this "2-sigma" range are
c       then included in an averaging algorithm. If there are no pixels
c       within the given ranges, then the 4 neighbor average is calcu-
c       lated to correct for this "sharp-shot noise".
c       ****************************************************************
c
c       *** DEFINITIONS ***
c
c       fname       filename
c       comlin      comments line for file storage
c       dev         standard deviation
c       devo        dev converted for program's use
c       less        2 standard dev's below 1, expressed as %
c       more        2 standard dev's above 1, expressed as %
c       flag        filter iteration number
c      delta       indicates if a given pixel is within 2dev's
c       zee         summation of all gray levels within 2 dev's
c       r           gray value of central pixel in 5x5 array
c       small       lower 2-sigma limit
c       large       upper 2-sigma limit
c       up          pixel directly above central pixel
c       dn          pixel directly below central pixel
c       g(x,y)      filtered pixel
c       spklindx    speckle index
c       trash       thresholding subroutine
c       speckle     speckle index subroutine
c
c       *** DECLARATIONS ***
c
        implicit integer*2 (a-z)
        integer*2 delta,errval,small,large,up,dn,zee,yy,xx,flag
        integer*2 q(5)
        real*4 dev,devo,less,more,spklindx
        character*127 comlin
        character*20 fname
        integer*2 p(512,240)
c
c       *** SET HARDWARE REGISTERS, INDICATE USE OF A PSEUDOCOLOR BOARD
c       AND INITIALIZE LUT's TO STANDARD VALUES ***
        call SETHDW(16#FF00,16#C000,1)
        call INITIA
```

```
      write(*,*) 'enter the name of the file to retrieve:'
      read(*,5401) fname
 5401 format(a)
      errval=READFT(0,0,512,512,fname,comlin)
      write(*,*) 'error code from READFT=',errval
      if (errval .eq. 0) then
              write(*,*) comlin
      endif
c
      call TRASH
c
      write(*,*) 'enter the standard deviation of the image:'
      read(*,*) dev
      devo=dev/256.
c
c     *** COMPUTE SPECKLE INDEX ***
c
      call SPECKLE(spklindx)
c
      less = 1-2*devo
      more = 1+2*devo
c
c     **************************************
c         This is the actual sigma filter
c     **************************************
c
      flag = 0
 500  flag = flag+1
          do 40 y=2,240
              do 30 x=2,509
                  delta = 0
                  zee=0
                  r=RPIXEL(x,y)
                  small=nint(less*r)
                  large=nint(more*r)
                  do 20 k=y-2,y+2
                      call RHLINE(x-2,k,5,q)
                          do 10 l=1,5
                          if ((q(l) .ge. small) .and. (q(l) .le.
     \                     large)) then
                                  delta=delta + 1
                                  zee = zee + q(l)
                          end if
   10                 continue
   20             continue
c
c    *** SHARP SHOT NOISE IS CORRECTED ***
c
```

```
                if (delta .le. 2) then
                    up=RPIXEL(x,y-1)
                    dn=RPIXEL(x,y+1)
                    g(x,y)=(up+dn+q(2)+q(3)+q(4))/5
                    go to 30
                end if
                g(x,y)=zee/delta
   30       continue
   40   continue
c
c       *** WRITE FIRST HALF INTO FRAME-GRABBER ***
c
        do 60 y=2,240
            do 50 x=2,509
                call WPIXEL(x,y,g(x,y))
   50       continue
   60   continue
      write(*,*)' the first half is filtered'
c
c       *** FILTER THE SECOND HALF ***
c
        do 100 y=240,480
            do 90 x=2,509
                delta = 0
                zee=0
                r=RPIXEL(x,y)
                small=nint(less*r)
                large=nint(more*r)
                do 80 k=y-2,y+2
                    call RHLINE(x-2,k,5,q)
                        do 70 l=1,5
                        if ((q(l) .ge. small) .and. (q(l) .le.
     \                   large)) then
                                delta=delta + 1
                                zee = zee + q(l)
                        end if
   70               continue
   80           continue
c
c       *** SHARP SHOT NOISE IS CORRECTED ***
c
                yy=y-240
                if (delta .le. 2) then
                    up=RPIXEL(x,y-1)
                    dn=RPIXEL(x,y+1)
                    g(x,yy)=(up+dn+q(2)+q(3)+q(4))/5
                    go to 90
                end if
                g(x,yy)=zee/delta
   90       continue
  100   continue
```

```
c       *** WRITE SECOND HALF BACK INTO FRAME-GRABBER ***
c
            do 120 y=240,480
                yy=y-240
                do 110 x=2,509
                    call WPIXEL(x,y,g(x,yy))
  110           continue
  120       continue
        write(*,*) 'the second half is finished'
c
        call SPECKLE(spklindx)
        call TRASH
c
c       *** ADMINISTRATIVE ***
c
        write(*,*) flag,' =flag'
        write(*,*) 'do you wish to store the image? (0 for yes)'
        read(*,*) o
        if (o .ne. 0) goto 400
        write(*,*) 'enter filename you wish to store the image under'
        read(*,5401) fname
        write(*,*) 'enter any comment, terminated with a !:'
        read(*,5401) comlin
        errval=SAVEFT(0,0,512,512,1,fname,comlin)
        write(*,*) 'error code from SAVEFT=',errval
 400    if (flag .le. 10) goto 500
        stop
        end
```

APPENDIX D.

LOCAL STATISTICS FILTER PROGRAM

```
$include: 'itexpc.inc'
c       ****************************************************************
c           This is the local statistics filter. It computes the local
c       variance and mean and then uses these numbers to calculate an
c       estimate of the pixel value.
c       ****************************************************************
c
c       *** DEFINITIONS ***
c       fname       filename
c       comlin      comments line for file retrieval
c       sum         summation of gray levels for 5x5 local array
c       errval      error value for use with file storage
c       flag        filter iteration number
c       q           array to hold five pixel values
c       meanz       average of 5x5 local array
c       meanz2      meanz squared
c       sum2        summation of variances of local array
c       dev         standard deviation, entered by user
c       stddev2     dev squared
c       varx        variance of image without noise
c       varz        variance of image with noise
c       k2          k parameter
c       g(x,y)      filtered pixel gray values
c       trash       threshold subroutine
c       speckle     speckle index subroutine
c       spklindx    speckle index
c
c       *** DECLARATIONS ***
c
        implicit integer*2 (a-z)
        integer*2 sum,errval,flag,q(5)
        real*4 meanz,sum2,meanz2,varz,varx,stddev2,dev,k2
        character*127 comlin
        character*21 fname
        integer*2 g(512,402)
c
c       *** SET HARDWARE REGISTERS, INDICATE USE OF A PSEUDOCOLOR BOARD
c       AND INITIALIZE LUT's TO STANDARD VALUES ***
        call SETHDW(16#ff00,16#c000,1)
        call INITIA
c
        write(*,*) 'enter the name of the file you wish to call up:'
        read(*,5401) fname
        format(a)
        errval=READFT(0,0,512,512,fname,comlin)
        write(*,*) errval,'=error code from readft'
        if (errval .eq. 0) then
            write(*,*) comlin
        end if
        call TRASH
c
        write(*,*) 'enter the standard deviation of the image:'
        read(*,*) dev
        stddev2=(dev/256.)**2
        write(*,*) stddev2,'=stddev2'
        flag=0
c
c       *** COMPUTE THE SPECKLE INDEX ***
c
        call SPECKLE(spklindx)
c
        flag=flag+1
        sum=0.
        sum2=0.
        do 60 y=2,400
            do 50 x=2,509
c
c       *** COMPUTE THE LOCAL MEAN ***
c
                do 20 l=y-2,y+2
                    call RHLINE(x-2,l,5,q)
                    do 10 k=1,5
                        sum=sum+q(k)
   10               continue
   20           continue
                meanz=sum/25.
                sum=0.
C
C       *** COMPUTE THE LOCAL VARIANCE ***
C
                do 40 l=y-2,y+2)
                    call RHLINE(x-2,l,5,q)
                    do 30 k=1,5
                        sum2=sum2+(q(k)-meanz)**2
   30               continue
   40           continue
C
C       *** COMPUTE THE ESTIMATE OF g(X,Y) ***
C
                varz=sum2/25.
                sum2=0.
                meanz2=meanz**2
                varx=abs((varz+meanz2)/(stddev2+1)-meanz2)
                k2=varx/(meanz2*stddev2+varx)
c               g(x,y)=int(meanz+k2*(RPIXEL(x,y)-meanz))
c
   50       continue
   60   continue
c
c       *** WRITE g(X,Y) ARRAY INTO FRAME-GRABBER ***
c
        do 80 y=2,400
            do 70 x=2,509
                call WPIXEL(x,y,g(x,y))
   70       continue
   80   continue
        call SPECKLE(spklindx)
        call TRASH
c
        write(*,*) 'do you wish to save to a file? (0 for yes)'
        read(*,*) O
        if (O .NE. 0) goto 100
        write(*,*) 'enter the filename to store the image under:'
        read(*,5401) fname
        write(*,*) 'enter any comment, terminated with a !:'
        read(*,5401) comlin
        errval=SAVEFT(0,0,512,512,1,fname,comlin)
        write(*,*) errval, '=error code from SAVEFT'
  100   if (flag .lt. 10) goto 5
c
        stop
        end
```

APPENDIX E.

GEOMETRIC FILTER PROGRAM

```
$include:'itexpc.inc'
c       ****************************************************************
c           This is a program to filter an image through the use of a
c       geometric algorithm. It calls subroutine 'speckle' to find the
c       speckle index, and subroutine trash to threshold the image. The
c       ITEXPC software is used extensively.
c       ****************************************************************
c
c       *** DEFINITIONS ***
c
c       fname       filename
c       comlin      comments line
c       flag        iteration number of filter
c       t           a counter; determines which quarter of image
c       w           row number
c       a,b         used to control direction within algorithm
c       c           used to determine direction
c       d           controls whether image or complement is filtered
c       spklindx    speckle index
c       q           array to hold a complete row of pixel gray levels
c       errval      error value for use in file storage
c       p(x,y)      image pixel gray levels
c       g(x,y)      image complement pixel gray levels
c       trash       threshold subroutine
c       speckle     speckle index subroutine
c
c       *** DECLARATIONS ***
c
        implicit integer*2 (a-z)
        integer*2 errval,flag,q(512)
        real*4 spklindx
        character*127 comlin
        character*20 fname
        integer*2 p(120,512),g(120,512)
c
c       *** SET HARDWARE REGISTERS, INDICATE USE OF PSEUDOCOLOR BOARD
c       AND INITIALIZE LUT's TO STANDARD VALUES ***
        call SETHDW(16#ff00,16#c000,1)
        call INITIA
c
        write(*,*) 'Enter the name of the file to retrieve:'
        read(*,5401) fname
 5401   format(a)
        errval=READFT(0,0,512,512,fname,comlin)
        write(*,*) 'error code from READFT = ',errval
        if (errval .eq. 0) then
           write(*,*) comlin
        end if
        call TRASH
        flag=0
  500   w=0
        e=0
        write(*,*)flag,'is the run number'
        flag=flag+1
c
c       *** COMPUTE THE SPECKLE INDEX ***
c
        call SPECKLE(spklindx)
        do 1500 t=1,4
c
c       *** CONVERT THE INPUT IMAGE INTO A 512x120 ARRAY, P(X,Y) ***
c
        do 2 y=1,120
            call RHLINE(0,w,512,q)
            w=w+1
            do 1 x=1,512
                p(y,x)=q(x-1)
   1        continue
   2    continue
c
c       *** THIS IS THE START OF THE GEOMETRIC FILTER ***
c
        a=1
        b=0
        c=3
        d=1
        write(*,*)'working. . . .'
  102   do 120 y=1,120
           do 110 x=1,512
                g(y,x)=max0(p(y,x),min0(p(y-a,x-b)-1,p(y,x)+1))
  110      continue
  120   continue
        do 140 y=1,120
           do 130 x=1,512
                p(y,x)=max0(g(y,x),min0(g(y-a,x-b),g(y,x)+1,
     \          g(y+a,x+b)+1))
  130      continue
  140   continue
        if (d .eq. 1) then
           a=-a
           b=-b
           d=0
           go to 102
        end if
```

```
      if (d .eq. 0) then
         d=1
      end if
105   do 160 y=1,120
         do 150 x=1,512
              g(y,x)=min0(p(y,x),max0(p(y-a,x-b)+1,p(y,x)-1))
150      continue
160   continue
      do 180 y=1,120
         do 170 x=1,512
              p(y,x)=min0(g(y,x),max0(g(y-a,x-b),g(y,x)-1,
    /         g(y+a,x+b)-1))
170      continue
180   continue
      if (d .eq. 1) then
         a=-a
         b=-b
         d=0
         go to 105
      end if
      if (d .eq. 0) then
         d=1
      end if
      if (c .eq. 3) then
         a=0
         b=1
         c=2
         go to 102
      end if
      if (c .eq. 2) then
         a=1
         b=1
         c=1
         go to 102
      end if
      if (c .eq. 1) then
         a=1
         b=-1
         c=0
         go to 102
      end if
c
c     *** WRITE BACK INTO FRAME-GRABBER AND DISPLAY ***
c
999   do 301 y=1,120
          do 300 x=1,512
              q(x-1)=p(y,x)
300       continue
              call WHLINE(0,e,512,q)
              e=e+1
301   continue
```

```
 1500   continue
c
c       *** ADMINISTRATIVE ***
c
        call TRASH
        write(*,*)'Do you wish to save image to a file? (0 for yes):'
        read(*,*) o
        if (o .ne. 0) goto 1501
        write(*,*)'enter the filename you wish to store the image under:'
        read(*,5401) fname
        write(*,*)'enter the comment, terminated with a "!" :'
        read(*,5401) comlin
        errval=SAVEFT(0,0,512,512,1,fname,comlin)
        write(*,*)'error code from SAVEFT = ',errval
 1501   if (flag .lt. 12) goto 500
        call SPECKLE(spklindx)
 997    stop
        end
```

LIST OF REFERENCES

1. Shook, M.P., Computer-Controlled Image Analysis of Solid Propellant Combustion Holograms Using a Quantimet 720 and a PDP-11, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1985.

2. Redman, D.N., Image Analysis of Solid Propellant Combustion Holograms Using an ImageAction Software Package, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1986.

3. Imaging Technology Incorporated, The ITEX/PC Programmer's Manual, Part Number 47-s00005-01, Revision 1.1, September 1985.

4. Edwards, T.D., Horton, K.G., Redman, D.N., Rosa, J.S., Rubin, J.B., Yoon, S.C., Powers, J.P. and Netzer, D.W., "Measurements of Particulates in Solid Propellant Rocket Motors," to be published in Proceedings of the 21st JANNAF Combustion Meeting, Chemical Propulsion Information Agency, Johns Hopkins University, Laurel, Maryland.

5. Netzer, D.W. and Powers, J.P., "Particle Sizing in Rocket Motor Studies Utilizing Hologram Image Processing," Proceedings of the NASA Workshop on Data Reduction From Images and Interferograms, NASA/Ames Research Center, January 1985.

6. Crimmins, T.R., "Geometric Filter for Speckle Reduction," Applied Optics, Vol. 24, No. 10, May 1985.

7. Crimmins, T.R., "Geometric Filter for Reducing Speckle," Optical Engineering, Vol. 25, No. 5, May 1986.

8. Lee, J.S., "Speckle Suppression and Analysis for Synthetic Aperture Radar," Optical Engineering, Vol. 25, No. 5, May 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor J.P. Powers, Code 62Po Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
5.	Professor D.W. Netzer, Code 67Nt Department of Aeronautics Naval Postgraduate School Monterey, California 93943-5000	2
6.	Commanding Officer Attention: Lieutenant M. Moser Air Force Rocket Propulsion Laboratory Edwards Air Force Base, California 93523	1
7.	Captain T.D. Edwards, USMC Marine Corps Tactical Systems Support Activity Camp Pendleton, California 92055-5080	2

GG

176 3/7
c

221002

Thesis
E256 Edwards
c.1 Implementation of three speckle reduction filters for solid propellant combustion holograms.

221002

Thesis
E256 Edwards
c.1 Implementation of three speckle reduction filters for solid propellant combustion holograms.