May 1988

Generalized Image Translation and Realignment: The GITR Process

Scott H. Novack
University of Pennsylvania

Generalized Image Translation and Realignment: The GITR Process

Abstract

Image processing is continually hampered by the effects of noise. This paper introduces the GITR noise reduction and analysis system. It is performing removal and analysis of noise in images of a scale model of the University of Pennsylvania campus. GITR uses several algorithms including voting and median filtering to remove noise and difference of images and histogram functions for noise analysis.

Also, this work includes reports on inherent noise contained in the cameras available in the Penn GRASP Laboratory. A stuck-pixel map and analysis of any inherent noisy screen pattern is presented for four camera systems. Statistical data for all noise patterns is also given.

Future research directions concerning other noise removal algorithms and comparisons with this study, and reports of noise profiles of other cameras or camera lenses are motivated by the results of this experiment, namely study of the effect of combining using rotated or scaled data.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-29.

GENERALIZED IMAGE TRANSLATION AND REALIGNMENT: THE GITR PROCESS

Scott H. Novack

MS-CIS-88-29
GRASP LAB 139

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

May 1988

Acknowledgements: This work was in part supported by: Air Force grant AFOSR F49620-85-K-0018, DARPA/ONR grant N0014-85-K-0807, NSF grant DCR-8410771, IRI84-10413-AO2, NSF-CER/DCR82-19196 A02, Army/DAAG-29-84-K-0061, DAA29-84-9-0027, NIH grant NS-10939-11 as part of Cerebrovascular Research Center, NIH 1-RO1-NS-23636-01, NSF INT85-14199, NSF DMC85-17315, by DEC Corp., IBM Corp. and LORD Corp.

UNIVERSITY OF PENNSYLVANIA

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

MOORE SCHOOL OF ELECTRICAL ENGINEERING

GENERALIZED IMAGE TRANSLATION AND REALIGNMENT:

The GITR Process

Scott H. Novack

Philadelphia, Pennsylvania
April 26, 1988

A report presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering for undergraduate work in the Department of Computer Science and Engineering.

Advisor: Dr. Ruzena Bajcsy

Abstract

Image processing is continually hampered by the effects of noise. This paper introduces the GITR noise reduction and analysis system. It is performing removal and analysis of noise in images of a scale model of the University of Pennsylvania campus. GITR uses several algorithms including voting and median filtering to remove noise and difference of images and histogram functions for noise analysis.

Also, this work includes reports on inherent noise contained in the cameras available in the Penn GRASP Laboratory. A stuck-pixel map and analysis of any inherent noisy screen pattern is presented for four camera systems. Statistical data for all noise patterns is also given.

Future research directions concerning other noise removal algorithms and comparisons with this study, and reports of noise profiles of other cameras or camera lenses are motivated by the results of this experiment, namely study of the effect of combining using rotated or scaled data.

Contents

1 Introduction

Image processing is continually hampered by the effects of noise. Regardless of purpose, it is obvious that a less noisy image is a more useful image. For instance, getting information extracted from an aerial photograph [HA 87] can be severely hampered by noisy picture elements or *pixels*—finite-sized cells of constant gray level that partition an image [BB 82, p.37]—that cut across edges or other features of the ground objects. Noise pixels are random variations in an image that cause undesirable effects [BB 82, p.65]. For instance, a white dot on one face of an otherwise gray cube would be a noisy pixel. Noisy pixels cause errors in further processing. For example, thinned edges from Canny's method [JC 86] may be broken or extended in aberrant directions. It would be beneficial to remove as much noise as possible. The first thing is to understand the noise, then it will be easier to eliminate, but exactly what procedures do a good job? And what kind of noise will they find to remove? Thus, the GITR (**G**eneralized **I**mage **T**ranslation and **R**ealignment) Experiment on noise reduction is presented.

This process was developed for use on 512 $\times$ 512 $\times$ 8 bits (256 gray levels) gray scale images and thresholded, thinned, gradient edge images of any scene with complex discernible features (not simple images). The process is not a real-time process, but rather useful for off-line analysis and camera evaluation. The system consists of modules for correlation, combination, and noise analysis which attempt to reduce and analyze noise without destroying or altering the original image as convolution algorithms would [BB 82].

Main objectives are:

- Use of several different techniques in the GITR noise reduction system.
- Analysis of techniques to deduce which one produces the highest "improvement."
- Portability of process to different camera systems in the Penn GRASP Laboratory.
- Construction of a stuck-pixel map and temporal pattern profile for all camera systems.

The following assumptions are made:

- Images are not highly textured.
- Images contain many edges <u>on every scanline</u>. (Figure 1)
- Images will be in focus to facilitate sharp edges.

- Actual contents of the scenes and illumination of the scenes are constant across the input time of the experiment.
- Images will be dissimilar within a limited range of displacements.
- Images will not be disparate due to stereo effects.
- Two image sets will be used to verify spatial noise analysis.

2 What is GITR?

GITR is basically an experiment in noise understanding. GITR attempts to reduce both spatial and temporal noise through the combining of several (between 5 and 8) images of the same object. Successive images are taken with a small translation in the camera—a jitter. Such displacement will not result in significant stereo disparity, only a shift in the image from the reference position, assuming the objects recorded are not at largely different depths in the field of view. In this experiment, a percentage of 3 percent or less for the ratio between maximum depth difference of any two objects and maximum camera distance was found to be acceptable.

The translated images are correlated with the reference image so that re-alignment can occur and facilitate combining the several images into one result picture. The logic behind the GITR process is as follows:

1. by combining several images, *temporal* noise—noise occurring randomly in time like a magnetic fluctuation phenomenon—will be removed and replaced by "correct" data pixels. (Figure 2)
2. by jittering the camera and correlating the images, *spatial* noise—fixed in location in all images of every object taken with the same camera (stuck pixel), usually a result of the camera's digitizing chip—will be removed.

The result should be a cleaner reference image.

Once clean images are created, noise analysis starts. The differences between the new clean image and the input images are examined for bright spots which are questionable pixels. Later, stuck pixels are confirmed. In addition, the difference image is brightened and examined for any temporal pattern which should not be in the clean image, assuming enough input images were combined (Figure 3). The pattern found moves through the image, but is not a representation of how an arbitrary pixel changes with time. Dataflow for the experiment appears in Figure 4.

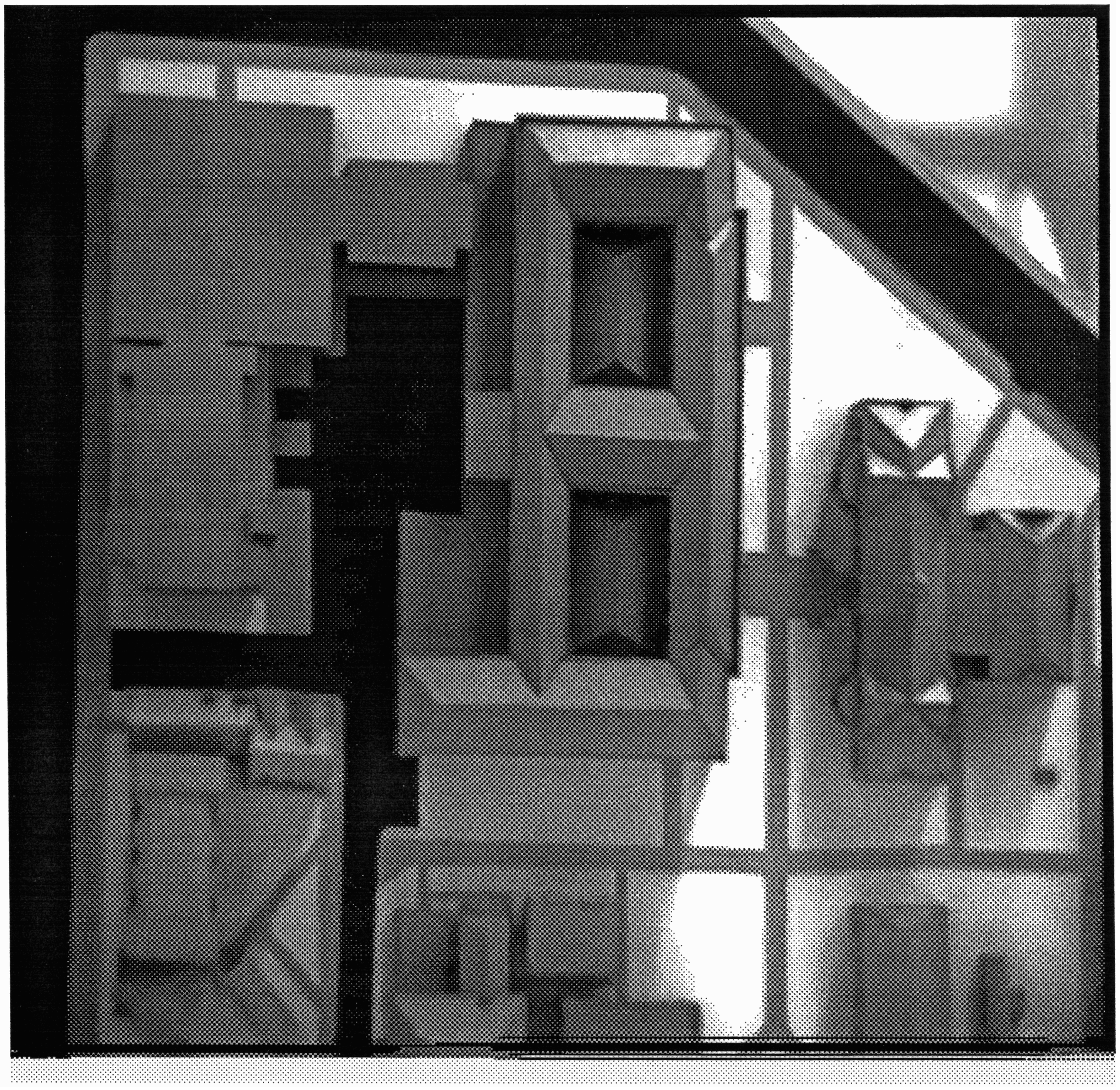

Figure 1: Typical GITR image.

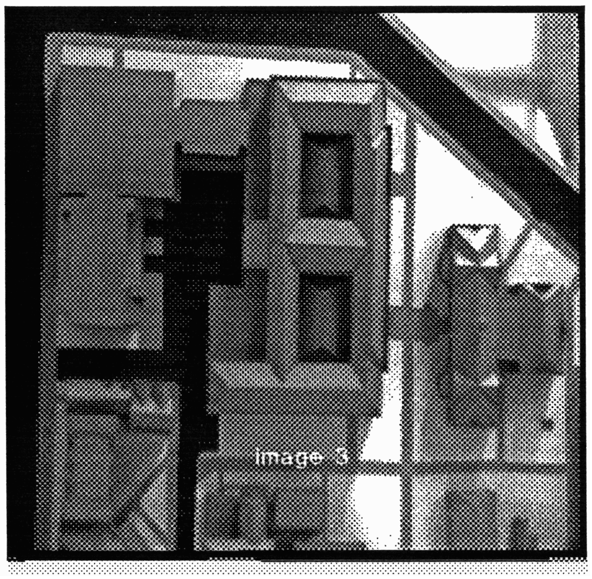

Figure 2: Deliberate noise in an image.

Definition of "Improvement"

In order to decide whether a pixel should be changed or not, it is necessary to define "improvement" of an image. The definition of improvement is that amount of an image which has been changed because it has been deemed "bad."

Some of the pixels that were changed would obviously be an improvement, but the white words in Figure 2, for instance, were *not* in the reference image. These pixels were correctly "voted out", but this does not improve the *reference* image. Including these pixels in the improvement count, would result in the amount of improvement in the *total* number of images rather than for one image. An average over the number of images could be taken, but, in the case of images like those in Figure 2, this average would not be representative of the improvement that might be rendered with images that don't contain obvious noise. It would be too high for a clean image.

The only important image is the reference image. Therefore, GITR uses the definition that improvement was defined only with respect to the reference image which was as clean as could be guaranteed. Therefore, the obvious noise of Figure 2 would not be counted, but any pixel that changed in the reference image would. Hence, improvement is the percentage of the reference image that was changed in the combining process.

Again, the question arises whether the changes made are, in fact, beneficial to the image. The logic for the combining process is sound enough (and will

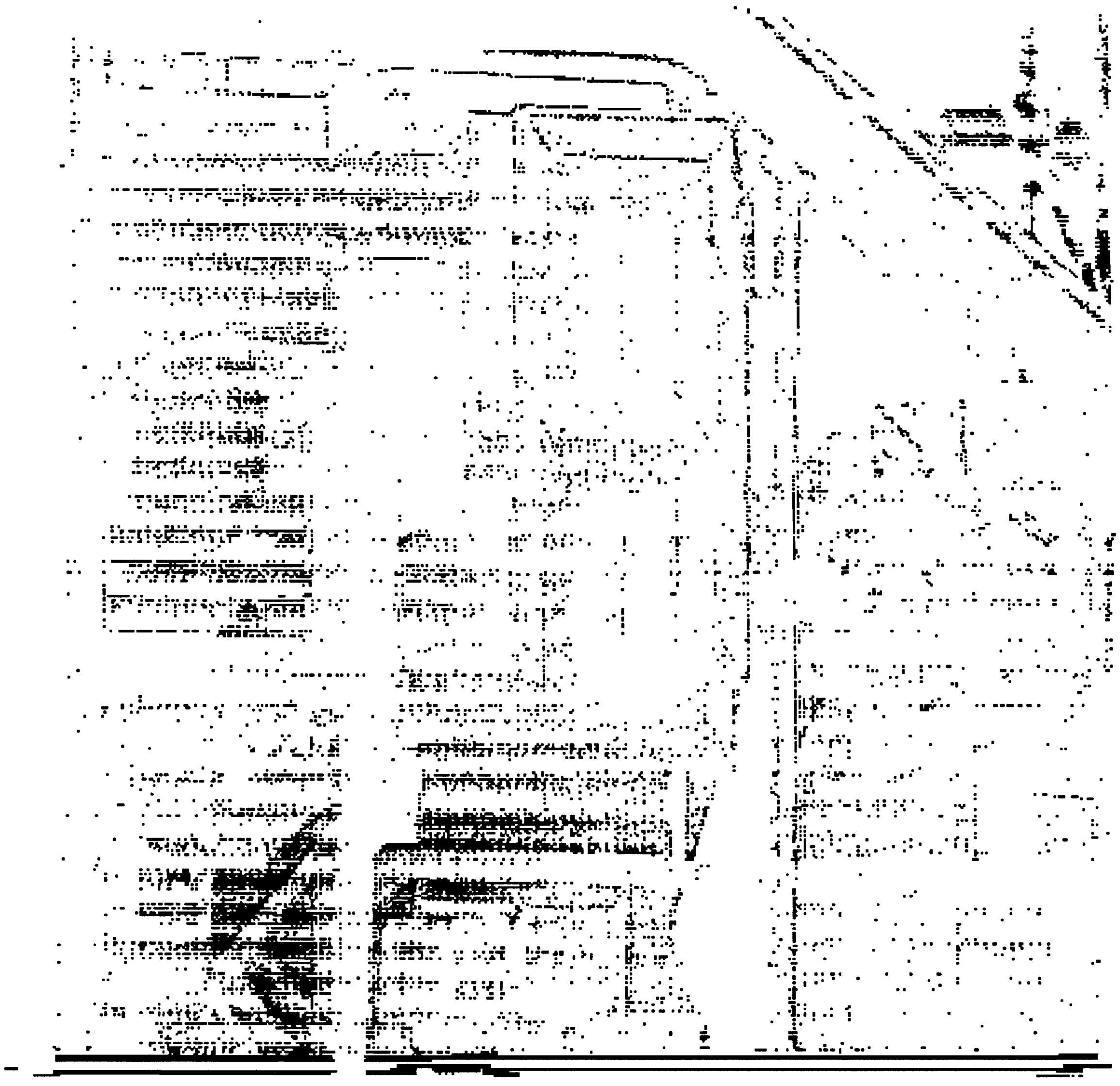
Figure 3: Typical screen pattern.

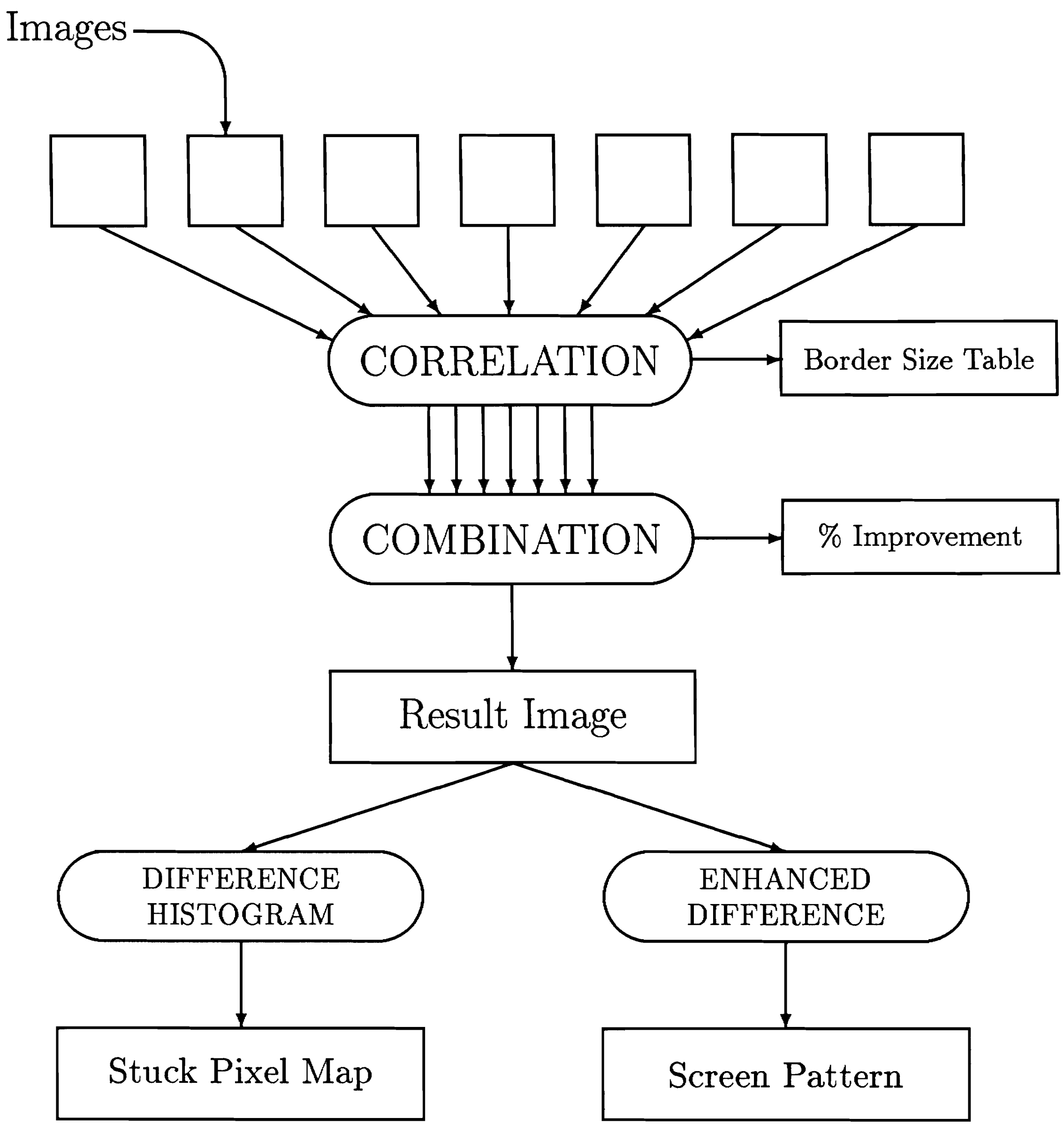

Figure 4: GITR Dataflow

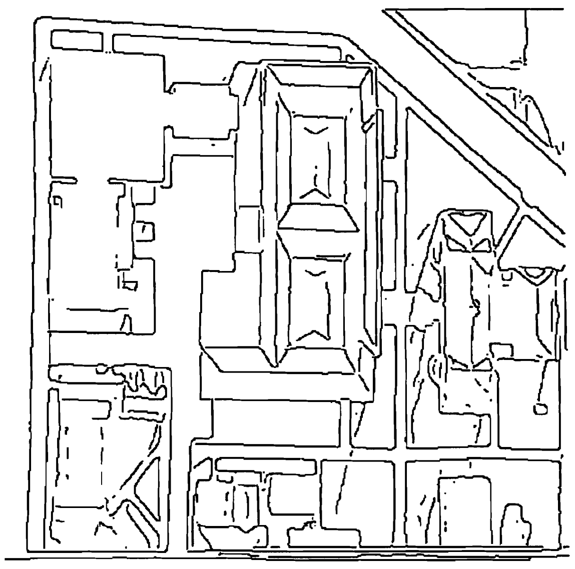

Figure 5: Image run through the Canny operator.

be described in Section 5), but the numbers returned for the improvement percentage are not conclusive in themselves. Just because algorithm A rendered a higher percentage improvement than algorithm B does not mean it performed better. It is indeed possible that the algorithm removed *too many* pixels in its attempt to remove noise. The algorithm cannot independently decide this since no *a priori* knowledge of what the final image should look like is input. For this reason, the various result images had to be personally inspected and a subjective judgment made. Since the human eye has trouble distinguishing pixels that are less than about 4 *gray levels* (quantized measurements of image brightness at a pixel [BB 82, p.23], 0 to 255 for the GITR images) apart in brightness [FVD 84, p.594], edge data were used to make judgments. Edge images are gray level images run through Penn's Canny operator which detects and highlights edges in the image (Figure 5). These pictures are much more easily compared. The results appear in Section 5.

4 Correlation

The following formula was used [BB 82, p.66]:

$$\mathbf{\Delta} = \sum(\mathbf{image1}[x][y] - \mathbf{image2}[x-dx][y])^2.$$

This formula is Euclidean distance squared. Delta is computed with different values of dx. The displacement at the minimum value for delta determines the best match, and, thus, the value for dx. A perfect match occurs when delta equals 0. The above formula's dx is the displacement in the second image in the x direction. This value is plugged into the dual of the formula to find delta-y. Then, this dy is plugged back into the original formula to see if the new y-displacement had any effect on the choice for dx. This process is reiterated until dx and dy do not change. For this project, displacements were kept between $\pm$ 20.

To demonstrate the correctness of the program, images taken a year ago as part of a project by Dominique Bartolo [DB 87] were used as input. The images were not translated from one another, but had different lighting. This correlation algorithm correctly deduced translations of 0 in both x and y for the images. The algorithm can be fooled however if the lighting creates a large amount of shadow and therefore a large amount of disparity even at no displacement. Consequently, a requirement for this experiment is that the images are constant in content *and illumination* over the period over which the images were taken. The algorithm will still perform well if the illumination requirement is violated, but not with the same degree of reliability.

To reduce computational complexity, calculations on only one *scanline* (horizontal or vertical line of pixels) in an image are performed. A problem with this technique is that if the image is featureless at that one scanline, no displacement will be discerned. This indeed occurred when taking correlation on images of an eye chart. For this reason, GITR requires images with abundant features on every scanline. This requirement could be avoided with certainty, but only with a drastic increase in computational complexity.

Another problem is that different camera systems have regions of "bad" data at the borders that are quite dissimilar. Sometimes this bad data can be composed of zeroes (black), or 255's (white). Sometimes it can be random values. Black regions do not effect the correlation algorithm, but white ones do. To counteract this effect, it is necessary to manually inspect the numbers being checked by the correlation algorithm and determine which ones are invalid. Then the correlation is changed to ignore the useless pixels and return a correct correlation. Each camera system must be treated separately. The results for the various cameras are in Table 1. Notice that a large amount of the right side of the Sony Laser Rangefinder images is unused.

One last feature of the algorithm exists. It is conceivable that a cyclic situation might arise during correlation. This means that the reiteration of the loop changed the dx value by one, which changed the dy value by one, which changed the dx value back, which changed the dy value back. This can happen if the aliasing of the image would place an image feature halfway between two pixel locations. This infinite loop is interrupted if it continues beyond ten iterations, and the algorithm will warn the user that the values

Camera	*# "bad" scanlines from*			
	left	*right*	*top*	*bottom*
Fairchild (red)	19	5	6	23*
Fairchild (green)	16	6	10	21
Sony	0	0	0	32
GE	4	0	0	33
Laser	8	161	2	32

*these pixels are 255; rest are 0.

Table 1: Border Size Table

have been cycling and therefore the final value may be off. It then suggests that the user consider redoing the trial without the problem image. If enough images remain, the output should not be adversely effected.

5 Combining

Once correlation data is determined for the set of images, it is used to combine the images into one clean image. Part of the experimental nature of this project is to find out how much noise several different combining algorithms would remove. The results are based on the percentage improvement in the final image over the initial image of the set (see Section 3). The algorithms tried were Voting, Median Filtering, Cedge and Cgray (courtesy of Dominique Bartolo [DB 87]), and combinations of these.

5.1 Voting

It has been said that the majority is always sane. It is assumed this applies to images as well. The idea behind voting is as follows:

1. Correlation data is used to access corresponding pixels in the various images.

2. The pixel in the first (reference) image is taken as the first "choice" of the final gray value for the result image since the result should be an improved version of the reference image. Any of the input images can be the reference image.

3. Other pixels are compared to Choice 1. If the difference between the test pixel and the reference pixel is not greater than the amount input by the

user—which is referred to as the *sameness value*—they cast a vote for Choice 1.

4. If it is greater, the first such pixel's value is deemed Choice 2. Any other pixels with equal or greater differences than Choice 2's cast votes for Choice 2.

5. At the end of the "election," the majority winner becomes the pixel in the result image. It is assumed that the other choice was the result of noise and should be ignored.

6. If Choice 2 is elected, then the reference image was wrong, and the improvement counter is incremented. A final percentage of the reference image that is changed is presented.

Voting was performed on two sets of data: gray level images and edge images. The gray level images were voted on with a sameness value of 20, and then the edge picture of the result image was produced. To compare with this, the edge pictures of all the input images were taken and combined into a result edge image. The sameness value of the edge combination was unimportant since the edge pictures have only black (0) and white (255) gray values.

The results were very similar, yet the differences are discernible. Combining the gray images produced an improvement of 2.62 percent as compared to 2.77 percent for combining the edge images. More important, however, is the judgment of how well they performed when looking at the final edge images from both trials. Gray combining improved the image less (by 0.14 percent), which would make it seem worse. It *is* worse as exhibited by the higher noise content of the image, which was small but could be made out. But it also did not remove too much. Looking at part of the edge combination result reveals that some of the "good" edges are incomplete whereas they are completely retained in the gray combination. Neither of these results is completely desirable, so it was obvious some other algorithm should be tested.

5.2 Median filtering

The logic of performing median filtering takes into consideration the Majority Rule idea of voting, but determines the choice gray level in a different way. The idea is as follows:

1. If there exists a majority of one value of pixel (or close to it) among the corresponding pixels of the images, this value will appear also in the median value.

2. If there is no real majority, chances are that noise will be lighter and darker (either extreme) than the real value which will appear in the median.

The implementation is:

1. Load all corresponding pixel values into a linear array.
2. Sort this array into either ascending or descending order. Any simple sorting algorithm will do because of the small set to be sorted. A shell-sort is used.
3. The middle element of the array is the final choice.

For this algorithm, a sameness value—acceptable difference between values of reference and test pixels—had no real impact on what final value was chosen. Instead, it was used to maintain consistency between median and voting trials. Sameness was used in median only to determine the amount of improvement.

With an identical sameness value, median filtering results fall in between those of the two voting techniques. Actual improvement was 1.03 percent. The final picture, on the other hand, was more satisfactory than the other two. Median found and kept more edges than edge combination and removed more noise than gray combination.

5.3 Other algorithms

The following algorithms were adapted, by addition of GITR's correlation algorithm, for use with this project.

5.3.1 Cedge

Cedge is an algorithm used previously by Dominique Bartolo to combine edges of the same object given different lighting conditions. It was not known whether this algorithm could be used with GITR. After extensive trials using all possible options of the program, it is concluded that cedge is not useful for this study. Cedge trials increased noise rather than removing it, for cedge thickens edges in an effort to connect them into whole regions. Option 5 of cedge had the thinnest edges, but even these were thicker than any edges produced from any of the other algorithms tried. There is no way to compare cedge results with the other results because of the disparity of the output data.

5.3.2 Cgray

Another of D. Bartolo's algorithms, Cgray, was tried also. Cgray was found to be unhelpful because it did not remove noise at all. It combined gray level images by using the maximum value of the corresponding pixels in the images. Thus, spatial noise was not only not removed, but also duplicated into its relative positions in the other images. In addition, the temporal noise in Figure 2, being white, stood out distinctly.

5.4 Combination of techniques

Combination of combining techniques occurred in two ways: using the median gray result image as reference image for gray voting, and using the edge image from the median result as reference for edge voting. The use of median as the first step will be explained in the next Section. The question is, if one algorithm improves the image, will another run of an improvement algorithm improve it appreciably? The answer is not really. Both algorithms rendered improvement of less than 1 percent (0.65 and 0.75 percent), but the author could not see any real change from the reference median image. Considering the processing time to run any of the combining algorithms, one combination is enough.

5.5 Comparison of algorithms

This study shows that median filtering is actually the best algorithm of those tried. Neither of the voting algorithms could be considered best since median combines the best attributes of both in one run. In addition, changing the sameness value from 20 to 10 caused some of the obvious noise in Figure 2 to appear in the result from voting. This occurred because of an underlying noise pattern which appears in the camera which is not immediately apparent. This error does not occur with median filtering since the noise value will almost assuredly not be found in the middle position. Also, median does not rely on order of images to come up with a value for Choice 2 as voting does; it is more concrete.

As the best algorithm, median is used as the base for combination of techniques and later for noise analysis. One warning: median can, and has, caused an error when used during spatial noise analysis, warranting the last assumption for the project (Section 1), although its overall performance is acceptable. The details appear in Section 6.1.

6 Noise analysis

Now that a reliable and clean result image has been generated, GITR focuses on the properties of the cameras that took the images. Although some noise present in images can come from outside (magnetic disturbances from low-flying planes or other nearby machinery), noise can also be inherent in the camera itself or from the digitizer connected to it. Errors in the digitizing chip of the system can cause certain pixels to be consistently incorrect or stuck at one value. This is spatial noise. The way the system is hooked up can cause a pattern to appear to scroll down the screen, affecting all the pixels over time. This is temporal noise. GITR strives to find any noise in the camera/digitizer system and report on its location (spatial) or waveform (temporal). It is important to know the characteristics of the noise for purposes of camera evaluation and system electrical configuration. The camera/digitizer systems used were:

1. Fairchild CCD stereo camera (red and green) with Penn's thumb dt2651 digitizer.
2. K. Wohn's Sony camera on the thumb digitizer.
3. GE camera from the GRASP Lab Laser Platform with Penn's index digitizer.
4. Sony Laser Rangefinder system on the index digitizer.

6.1 Spatial

Spatial noise comes in several varieties. The easiest to find is a pixel "stuck" on white or black. Then there are pixels that are not stuck, but are additive; that is, always x gray levels brighter/dimmer than the correct value. Finally, pixels can be at any random value.

6.1.1 Goals

The goal of spatial noise analysis is a map for each camera system of error pixels. Once these pixels are identified, it is very simple to preprocess any further images taken with these cameras so as to reassign a correct value to the problem pixels, perhaps assigning the median or mean value of the pixel's 8-connected neighbors. Since cameras are made with a certain degree of quality, the number of bad pixels should rarely be of a quantity to cause preprocessing to have a burdensome computational complexity. The result is that later users of the system need never worry about the problem pixels interfering with their research.

In addition, a look at the stuck pixel map of a camera system yields information useful for proposing to keep, fix, or replace the camera. Every once in a while, this test can be run on a camera to see if its internal workings are degenerating or if it is now unacceptable. This same data can be referred to when purchasing new cameras.

6.1.2 Implementation

To find noise pixels, first GITR uses an algorithm to produce the unsigned difference image between the median combination image and an arbitrary input image. Immediately viewing this image may show some bright spots which could be bad pixels. Bad pixels will be brighter than most because they are consistently off the true value.

To verify spots in the image, the difference image is passed into a histogram program with a user-input threshold value. This threshold value is a report value for the histogram. It is usually beneficial to start the threshold at the brightest gray level of the system. In this case, the histogram produces a list of how many pixels are at which gray level in the difference image. The histogram should be inspected for a value such that some, but not *too* many, pixels lie above this value. Then the difference image should be viewed thresholded at this value. Edges will appear from imperfect matching due to digitizer aliasing and focus. Also, some individual pixels should appear. If not, try a different value.

Once a satisfactory threshold is found, run the histogram again with this threshold value. The program will yield a list of the coordinates of all the pixels at or brighter than this threshold. The output of this program should be saved.

Repeat this sequence with all the input images. Each should be thresholded at the same level to retain consistency in finding bad pixels. Bad pixels may be off by the same amount in several images, and different thresholds might pass over them in some of the images. Once all the images have been processed, the histogram files are inspected. Bad pixels will appear at the same exact coordinates in several images. Edges may also appear, and the user must be careful not to determine that a long line of pixels that are off in all the images is a large amount of bad pixels. Most bad pixels will appear alone or, rarely, in pairs. Careful inspection of the histogram files will produce pixels that appear to part of no feature in the image, but are off nonetheless. These pixels appear in Table 2.

Notice how far inferior the red Fairchild camera (one of a stereo pair) is. It's first four pixels are additive, the last two are subtractive.

Camera	*# of pixels*	*Coordinates*
Fairchild (red)	6	(51, 75) (67, 65) (175, 35) (176, 35) (247, 198) (377, 205)
Sony	0	—
GE	0	—
Laser	0	—

Table 2: Stuck Pixels by camera.

One requirement of this test is that it should be run on more than one set of objects. This is to prevent pixel washout. Should a pixel be stuck on white, but the region it is in is white also, the pixel will fail to appear in the histogram survey. Another test will find this pixel (assuming the region it's in is not white again) and verify the data from the first trial.

Now the stuck pixel map can be used for preprocessing. However, a strange phenomena appeared in the red Fairchild camera. Upon processing another set of images, the "same" bad pixels were discovered in different locations. It is certain that these other pixels are the same ones because they were always the same relative distance from one another although the entire map had moved. It is possible that this occurs because of changes in the memory addressing in the digitizer chip. This camera will have to be checked every month or so to determine if it has again shifted it's pattern.

6.2 Temporal

Temporal noise is often more subtle than spatial noise. Noise like the words in Figure 2 is ridiculously blatant, but often the noise appears as an underlying pattern scrolling down the input and changing the value of any one pixel over time. This pattern may or may not be visible and can be any waveform.

6.2.1 Goals

The goal of this part of the experiment is to gather as much information as possible on any such pattern in the camera systems. Once this pattern is identified and measured, further simple tests can be run to determine the actual change in time of any pixel. The pattern shows how a pixel is going to change, but not in what time. Strict watch of time taken between image digitization must be kept to retrieve this information. For now, the information

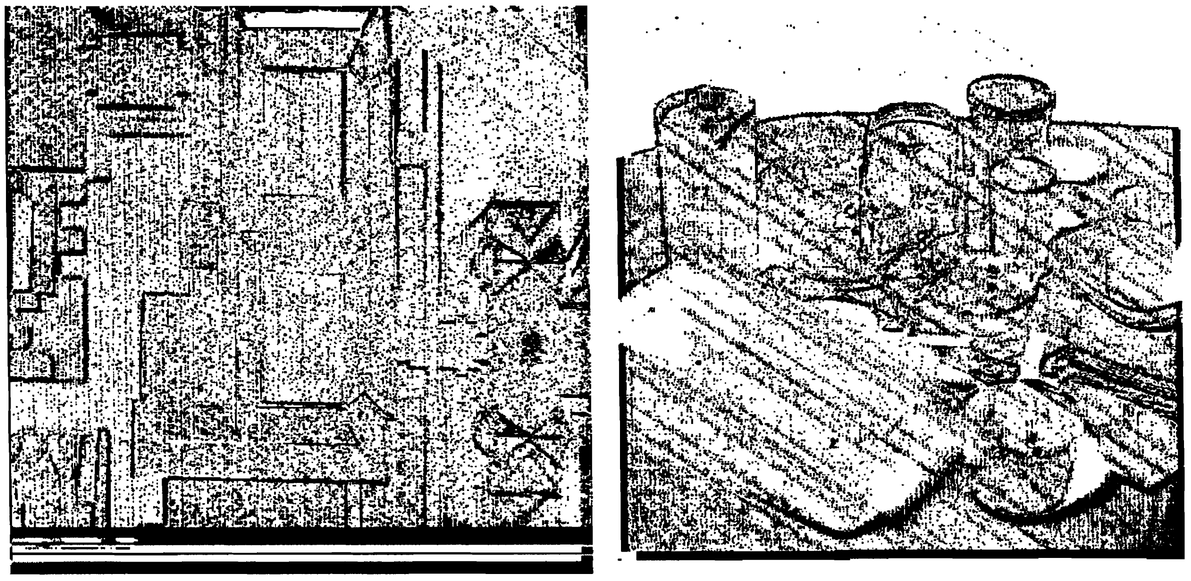

Figure 6: Vertical (a) and diagonal (b) noise patterns.

retrieved will be the pattern's waveform and this waveform's own amplitude and frequency (in pixels) along with statistical data.

6.2.2 Implementation

The implementation of the pattern search is similar to the spatial search. As before, bring up a difference image. Brighten (threshold) it at the value used in spatial analysis. Now brighten it further. Eventually, a pattern will be noticed like the one in Figure 3 or in another direction like vertical or diagonal (Figure 6).

The diagonal noise pattern is a strange occurrence because all the rest of the noise patterns are horizontal or vertical (See Figure 3, Figure 6a, and Appendix B). Conjecture on how this came about for this camera system (Fairchild green) amounts to coupling of the vertical and horizontal sync signals into the digitization of the image [GT 88]. The combination of the two signals would create a diagonal pattern with slope dependent on offset of the sync signals.

Detection is not a complete study of the noise patterns. Amplitude, waveform, and frequency are all of interest. Waveform for the various patterns was difficult to determine because of low amplitudes. For those patterns described as "bands", the waveform is probably a sinusoid or triangle wave caused by offset and overlap in the digitizer memory [JH 88] (Figure 7). Besides bands,

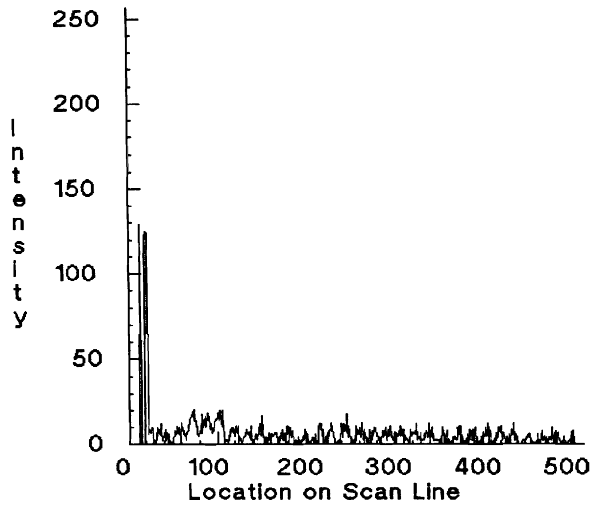

Figure 7: Waveform for Fairchild (red) noise pattern.

the remaining patterns are described as "lines" because of a width of 1 pixel. Frequencies are all peak to peak. Frequency information was gained through the use of scanline intensity plotter and image rotation programs.

Fairchild (red):	horizontal bands. Freq. ≈10-15 pixels.
Fairchild (green):	diagonal bands. Frequency 10-15 pixel equivalent.
Sony:	vertical lines. Frequency of 1 line per 3-4 pixels.
GE:	horizontal bands. Frequency ≈ 4 pixels.
Laser:	vertical lines. Frequency of 1 line per 3-4 pixels.

Amplitudes for the various patterns are described with statistical data. This data—variance, standard deviation, and mean—was computed with another algorithm described in Appendix A. Standard deviation and mean data appear in Figure 8. Notice that the Fairchild cameras are not only inferior with respect to bad pixels (Table 2), but also below par with respect to amplitude of the noise pattern.

The computation algorithm needs a threshold value which marks the boundary between background pattern noise and noise from edges or bad pixels in the difference image. A suitable threshold can be determined during brightening the difference image. When the noise pattern is discovered, that value will be the computational threshold. Edges and bad pixels must be ignored to get statistical data on only the pattern (Figure 9).

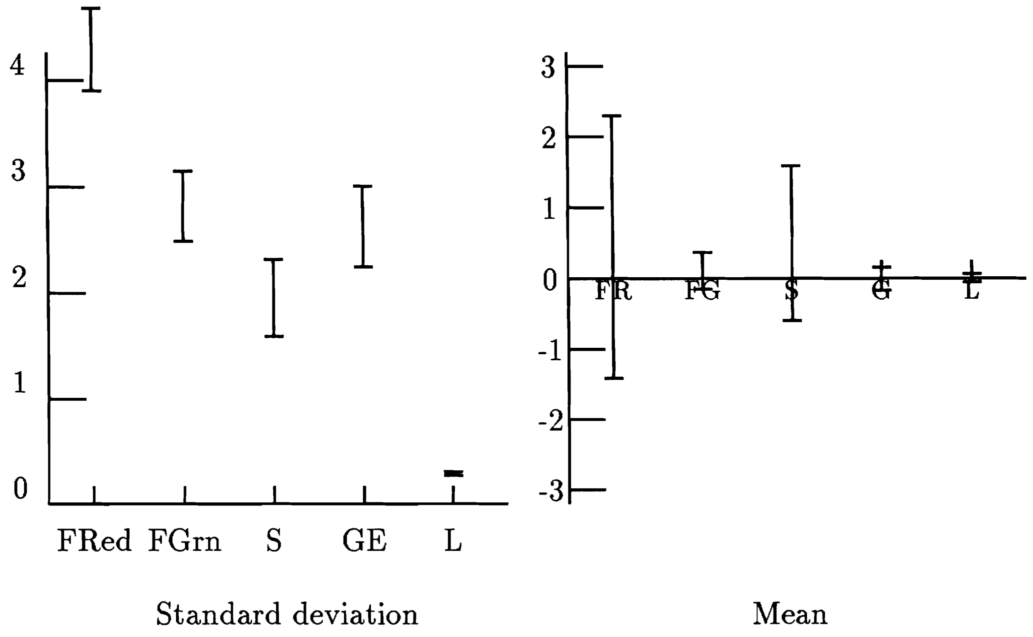

Figure 8: Statistical information by camera system.

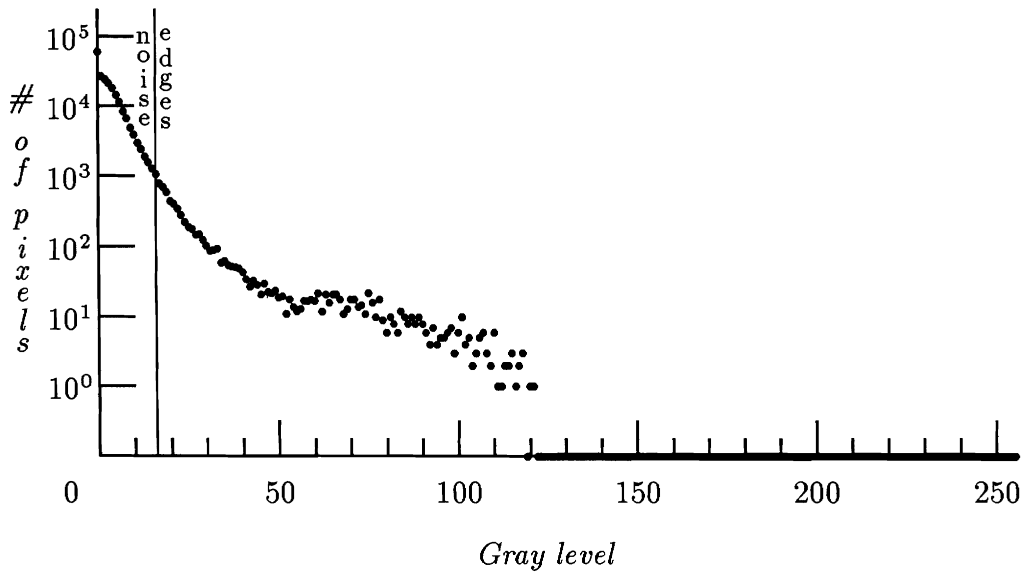

Figure 9: Histogram of a typical difference image.

7 Conclusions

The GITR Experiment is a success. GITR improves images using median filtering over other combining methods. As further research in the area of image improvement through combining, the author proposes using rotations or scaling as basis for combining: GIRR—Generalized Image Rotation and Realignment; ISMI—Image Scaling, Matching, and Interpolation.

GITR also serves as camera/digitizer evaluator. GITR finds bad pixels such that preprocessing will forever remove this bad data from consideration. GITR also discovers noise patterns involved in the system. With all this data, the research facility owning the cameras can make intelligent proposals concerning further camera purchases or replacements. For instance, the Fairchild camera used in this experiment is the most unreliable, especially the red. This is compounded by the fact that it is supposed to give stereo data. The replacement or repair of the Fairchild cameras should be considered. Other uses for the Sony cameras should also be considered in the light of this experiment.

A Shell Script

The following is provided as a tool for actually performing GITR on a new or old camera system on a UNIX machine with an IKONAS monitor:

WARNING It is assumed that the border information is already known for the camera system and the correlation algorithm changed accordingly.

Use the median filter combining program. Vary sameness at will.

```
% med -n <# images> -v <sameness val> -o <out_name>
     -i <in_name> -i ... > <datafile>
% diff <out_name> <any in_name> <diff_name>
% cp <diff_name> <dummy_name>
% hist -i <dummy_name> -o <hist_name> -v 250
```

This will yield, improvement data, correlation data, and a histogram without any stuck pixels. The first histogram should now be inspected and a threshold chosen such that there are some, but not too many points above the threshold. If possible, the difference image should be viewed when thresholded at this value. Edges will appear from imperfect matching due to digitizer aliasing and focus. Also, some individual pixels should appear. If this is not true, try a different threshold until one is satisfactory.

```
% hist -i <dummy_name> -o <hist_name> -v <threshold>
```

The diff and hist parts are run for each image. Then, the various histogram files are visually inspected and pixels appearing in all, or most, histograms are logged as stuck pixels. Now get the screen pattern:

```
% ikload <diff_name>
% bright <some val>
% bright <some lower val>, etc.
```

This will eventually display the pattern. To analyze it, various scanline plots and rotations can be run. Finally, to get the variance, standard deviation, and mean of the noise (x and y are the correlation values computed earlier):

```
% comp -i <imj> -m <median imj> -d <diff imj> -v <threshold>
      -c x -c y > <file>
```

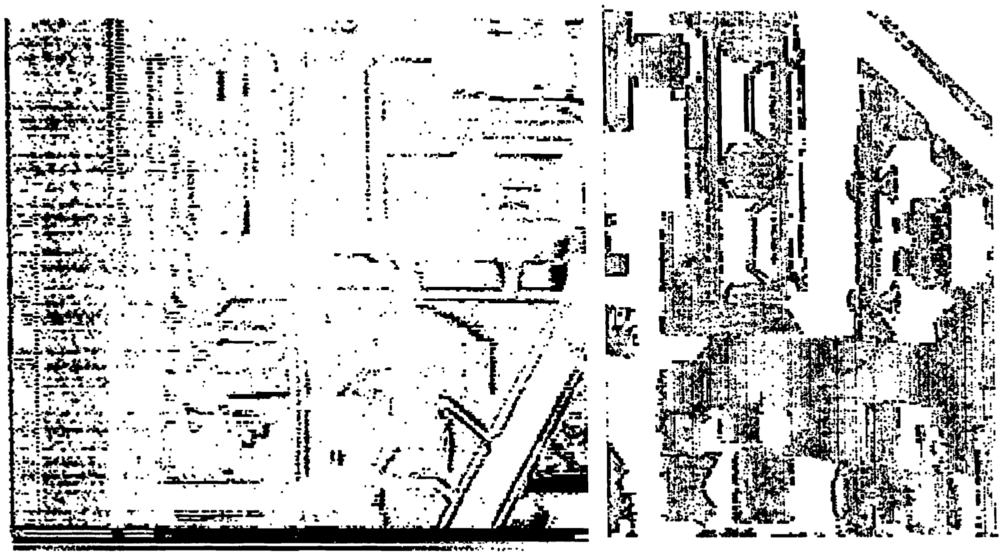

Figure 10: Noise patterns for GE camera (**a**) and Laser Rangefinder (**b**).

B Other Relevant Images

References

[HA 87] H.L. Anderson. "Edge Detection for Object Recognition in Aerial Photographs," University of Pennsylvania GRASP Laboratory Technical Report 96, 1987.

[BB 82] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[DB 87] Dominique Bartolo, et al. "Eliminating Shadows and Reducing Noise by Combining Pictures with Different Lighting Directions," University of Pennsylvania, August, 1987.

[JC 86] J. Canny. "A Computational Approach to Edge Detection," in *IEEE Trans. on PAMI*, Volume PAMI-8, Number 6, November, 1986, pp. 679-698.

[FVD 84] J.D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, Massachusetts, 1984.

[JH 88] John Hoford, personal conversation.

[GT 88] Gus Tsikos, personal conversation.