

10.6 Nonlinear Averaging

The linear averaging filters discussed so far blur edges. Even worse, if the mask of the smoothing operator crosses an object edge it contains pixels from both the object and the background, giving a meaningless result from the filter. The same is true if averages are performed when a certain number of pixels in an image show erroneous values, e. g., because of a transmission error. The question, therefore, is whether it is possible to perform an averaging that does not cross object boundaries or that ignores certain pixels. Such a procedure can only be applied, of course, if we have already detected the edges or a distorted pixel.

In this section, we discuss three types of nonlinear averaging filter: the classical median filter (Sect. 10.6.1); weighted averaging, also known as normalized convolution (Sect. 10.6.2); and steerable averaging (Sect. 10.6.3), where we control the direction and/or degree of averaging with the local content of the neighborhood.

10.6.1 Median Filter

Linear filters effectively suppress Gaussian noise but perform very poorly in case of binary noise (Fig. 10.7). Using linear filters that weigh and sum up, we assume that each pixel carries some useful information. Pixels distorted by transmission errors have lost their original gray value. Linear smoothing does not eliminate this information but carries it on to neighboring pixels. Thus the appropriate operation to process such distortions is to detect these pixels and to eliminate them.

This is exactly what a rank value filter does (Sect. 4.3). The pixels within the mask are sorted and one pixel is selected. In particular, the *median filter* selects the medium value. Since binary noise completely changes the gray value, it is very unlikely that it will show the medium gray value in the neighborhood. In this way, the medium gray value of the neighborhood is used to restore the gray value of the distorted pixel.

The following examples illustrate the effect of a 1×3 median filter \mathcal{M} :

$$\mathcal{M}[\dots 1\ 2\ 3\ 7\ 8\ 9\ \dots] = [\dots 1\ 2\ 3\ 7\ 8\ 9\ \dots]$$

$$\mathcal{M}[\dots 1\ 2\ 102\ 4\ 5\ 6\ \dots] = [\dots 1\ 2\ 4\ 5\ 5\ 6\ \dots]$$

$$\mathcal{M}[\dots 0\ 0\ 0\ 9\ 9\ 9\ \dots] = [\dots 0\ 0\ 0\ 9\ 9\ 9\ \dots]$$

As expected, the median filter eliminates runaways. The two other gray value structures — a monotonically increasing ramp and an edge between two plateaus of constant gray value — are preserved. In this way

What are the mean and median filters?

Mean filter

The mean filter is a spatial filter that replaces the center value in the window with the average of all the pixel values in the window. The window is usually square but can be any shape. An example of mean filtering in a 3x3 window is shown below.

unfiltered values		
5	3	6
2	1	9
8	4	7

$$5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45$$

$$45 / 9 = 5.$$

mean filtered		
*	*	*
*	5	*
*	*	*

Center value (previously 1) is replaced by the mean of all nine values (5).

Median filter

The median filter is also a spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window. The kernel is usually square but can be any shape. An example of median filtering of a single 3x3 window of values is shown below.

unfiltered values		
6	2	0
3	97	4
19	3	10

in order: 0, 2, 3, 3, 4, 6, 10, 15, 97

median filtered		
*	*	*
*	4	*
*	*	*

Center value (previously 97) is replaced by the median of all nine values (4).

Note that for the first (top) example, the median filter would also return a value of 5, since the ordered values are 1, 2, 3, 4, **5**, 6, 7, 8, 9. For the second (bottom) example, though, the mean filter returns the value 16 since the sum of the nine values in the window is 144 and $144 / 9 = 16$. This illustrates one of the celebrated features of the median filter: its ability to remove 'impulse' noise (outlying values, either high or low). The median filter is also widely claimed to be 'edge-preserving' since it theoretically preserves step edges without blurring. However, in the presence of noise it does blur edges in images slightly.

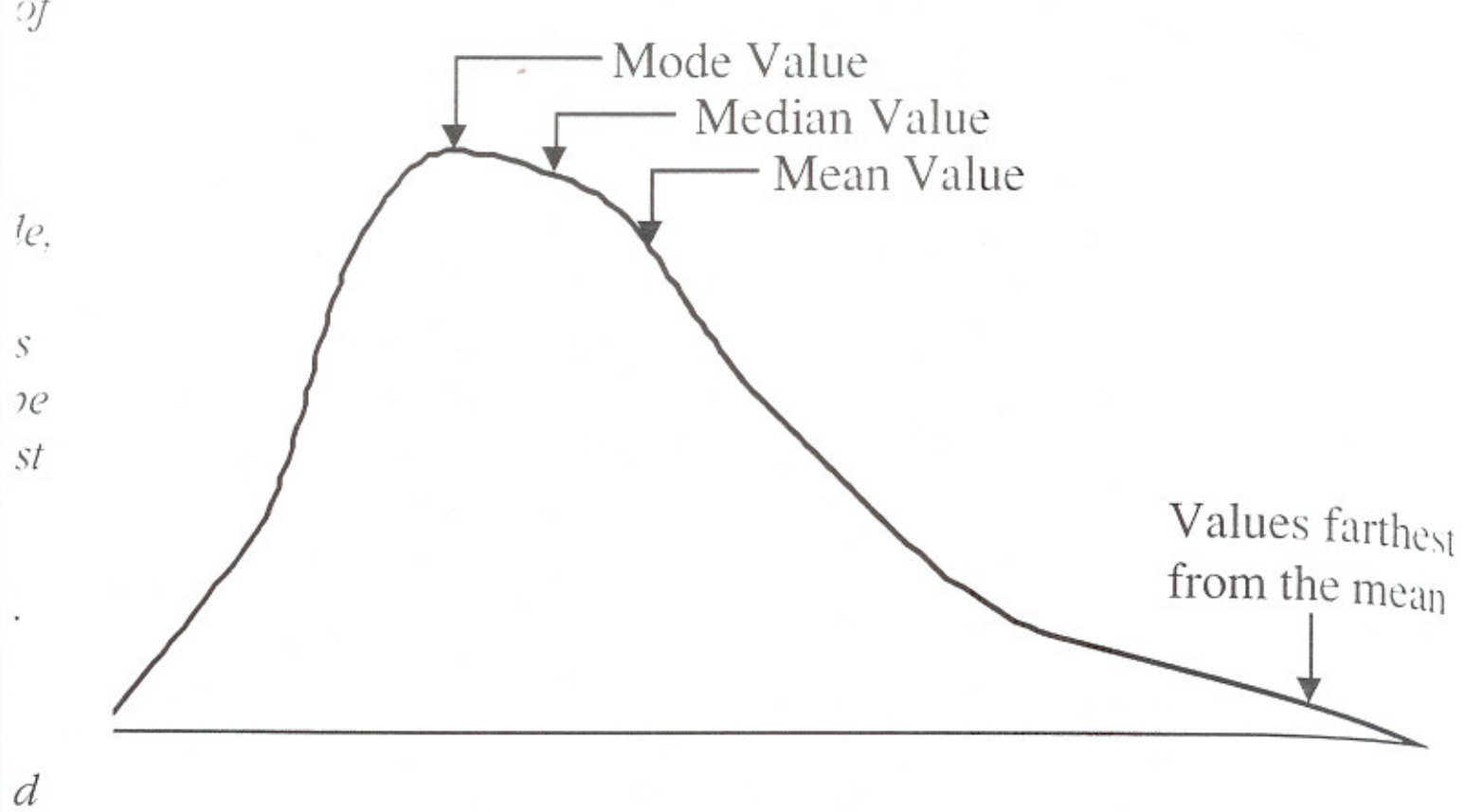
General Image Processing References

[Kenneth R. Castleman](#), *Digital Image Processing*. [Prentice Hall](#), 1996.

Anil K. Jain, *Fundamentals of Digital Image Processing*. [Prentice Hall](#), 1989.

William K. Pratt, *Digital Image Processing*. Wiley, 1991.

I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*. Kluwer Academic, 1990.



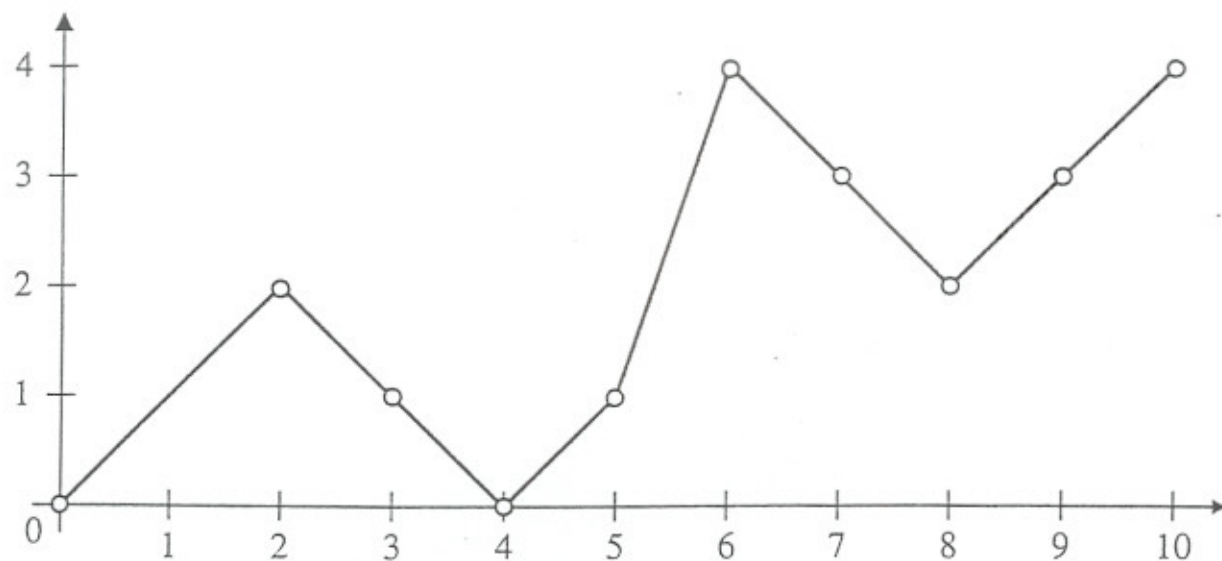
a 3×3 neighborhood by skipping the two pixels whose brightness values are most different from the mean, ranking the remaining seven values, and assigning the median to the central pixel. This has the effect of sharpening steps, and produces posterization when it is applied repeatedly.

Another modification to the median filter is used to overcome its tendency to erase lines which are narrower than the half-width of the neighborhood, and to round corners. The so-called hybrid median, or edge-preserving median, is actually a three-step ranking operation (Nieminen et al., 1987). In a 5×5 pixel neighborhood, pixels are ranked in two different groups as shown in **Figure 16**. The median values of the "X" and "+" groups (both of which include the central pixel) are compared to the central pixel and the median value of that set is then saved as the new pixel value. As shown in **Figure 17**, this method preserves lines and corners which are erased or rounded off by the conventional

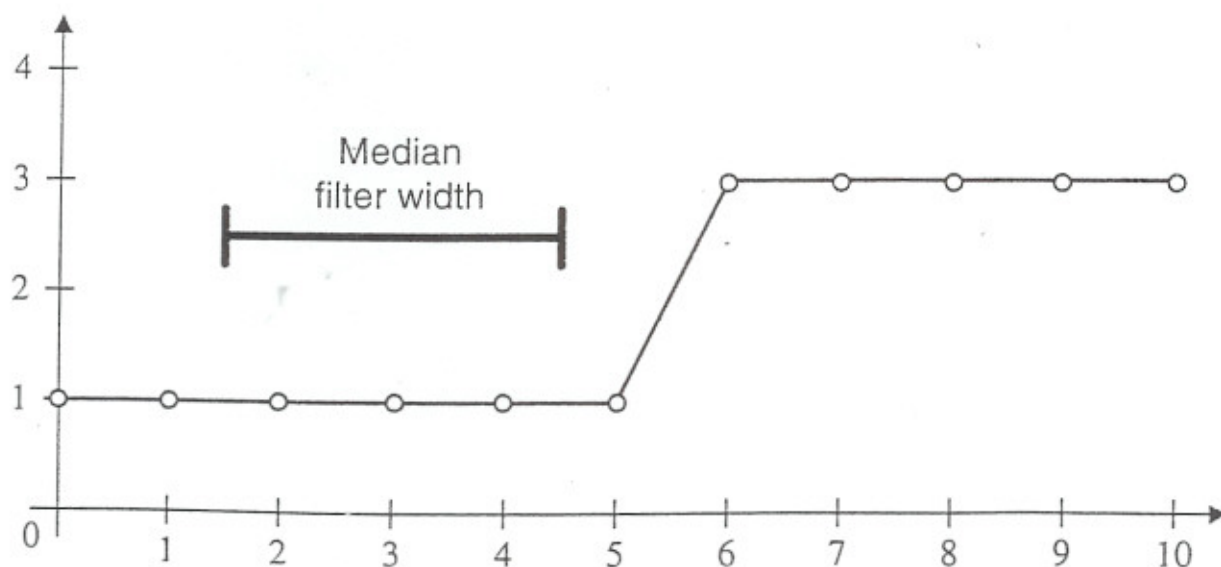
with the sampling frequency, and the median is computed over a three-point neighborhood. In this example, the median filter removes the sinusoid completely, while preserving the edge.

In general, light or dark objects having less than half the area of the median filter are completely eliminated, while larger objects are preserved approximately intact. Thus, the extent of the median filter must be "tuned" to the problem at hand. There is much less guidance for the design of median filters than there is to guide linear filter design. Experimentation often substitutes for analysis.

The noise-reducing effect that a median filter has on an image depends on two related, but usually separate, things: the spatial extent of the neighborhood (mask), as mentioned



(a)



(b)

Figure 11-33 Median filtering in one dimension: (a) input, (b) output

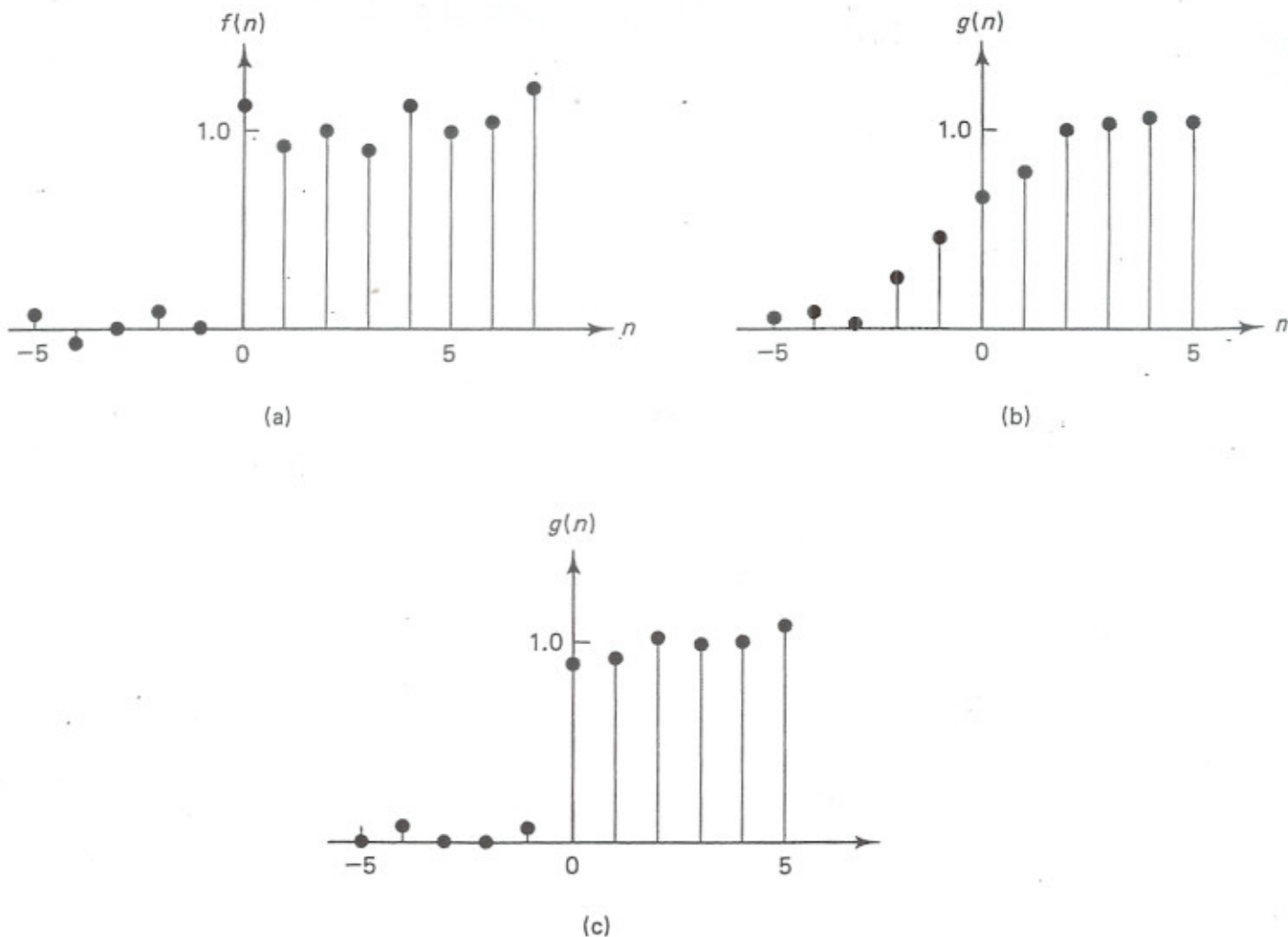


Figure 8.17 Illustration of median filter's tendency to preserve step discontinuities. (a) One-dimensional step sequence degraded by random noise; (b) result of lowpass filtering the sequence in (a) with a 5-point rectangular impulse response; (c) result of applying a 5-point median filter to the sequence in (a).

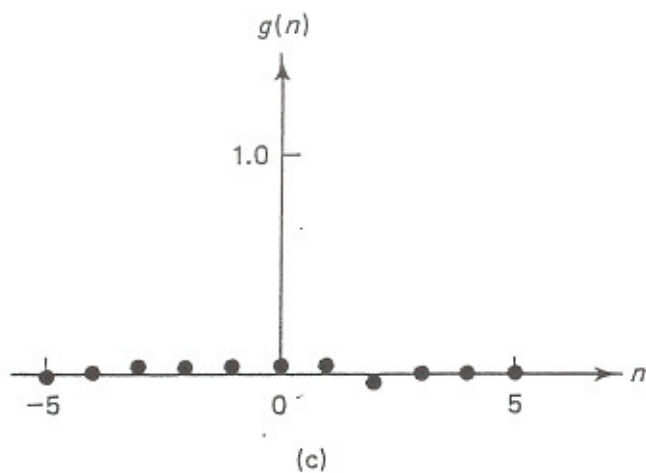
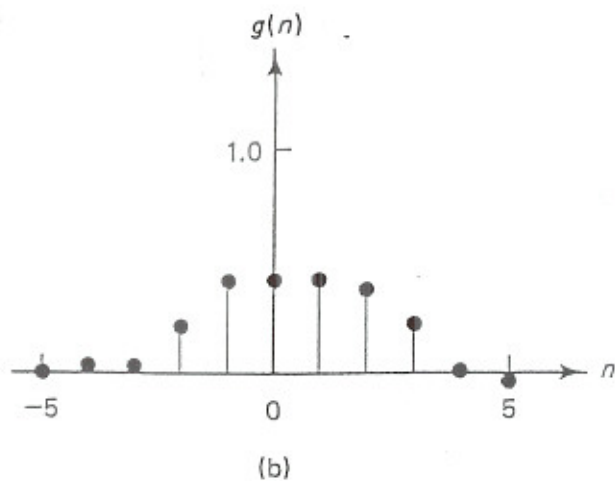
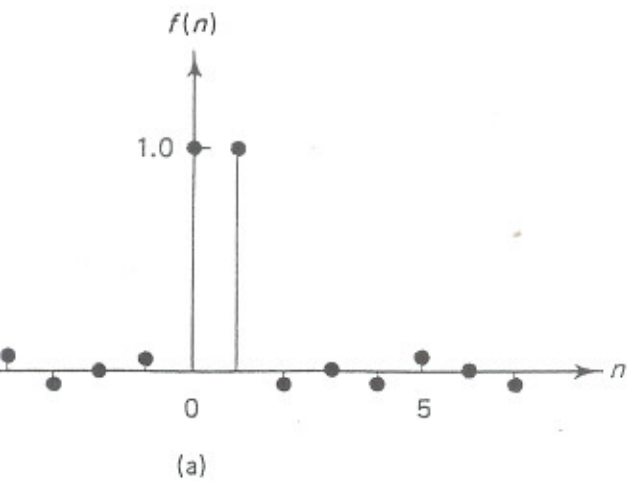


Figure 8.18 Illustration of a median filter's capability to remove impulsive values. (a) One-dimensional sequence with two consecutive samples significantly different from surrounding samples; (b) result of lowpass filtering the sequence in (a) with a 5-point rectangular impulse response; (c) result of applying a 5-point median filter to the sequence in (a).

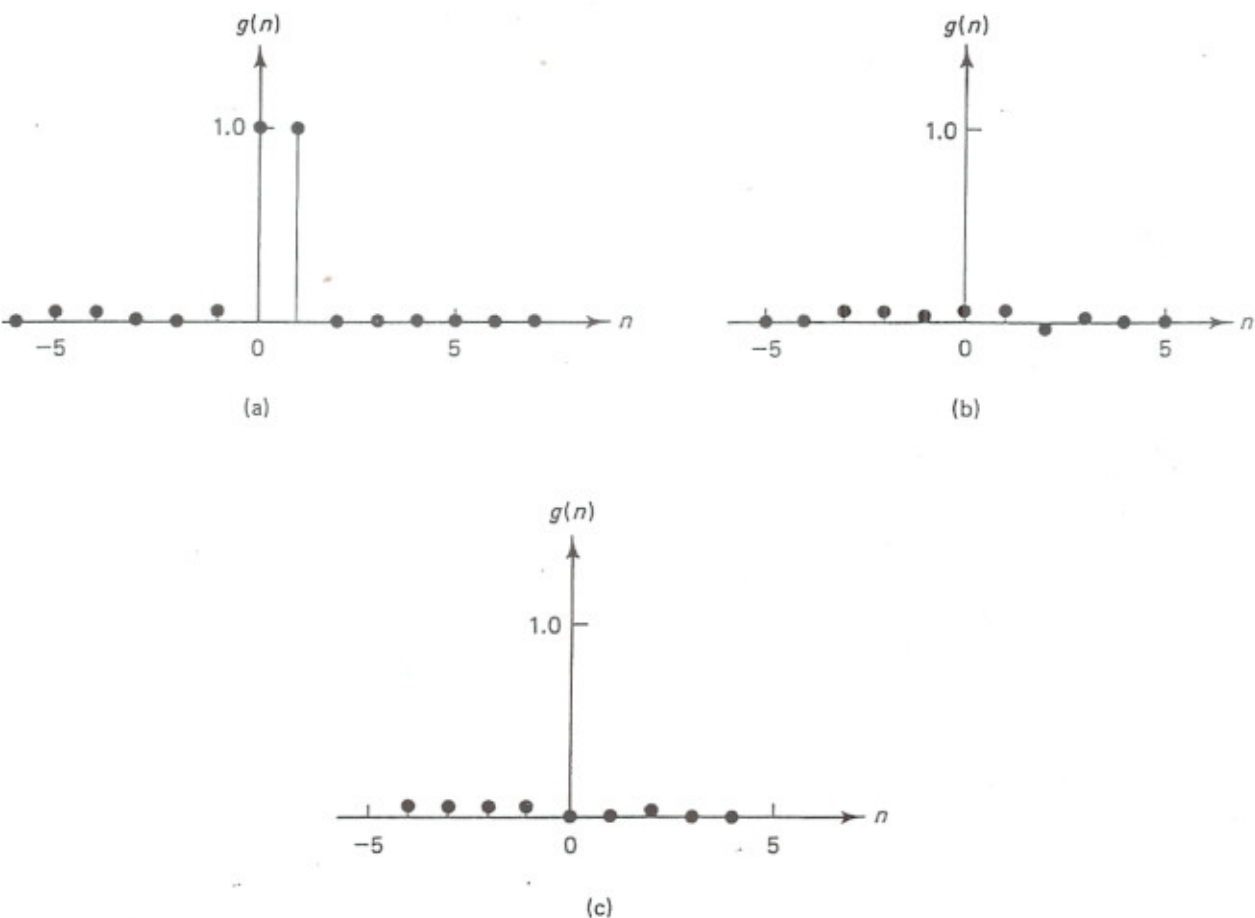


Figure 8.19 Results of applying a median filter to the sequence in Figure 8.18(a) as a function of window size. This illustrates that removal of impulsive values by a median filter depends on the window size. (a) Window size = 3; (b) window size = 5; (c) window size = 7.

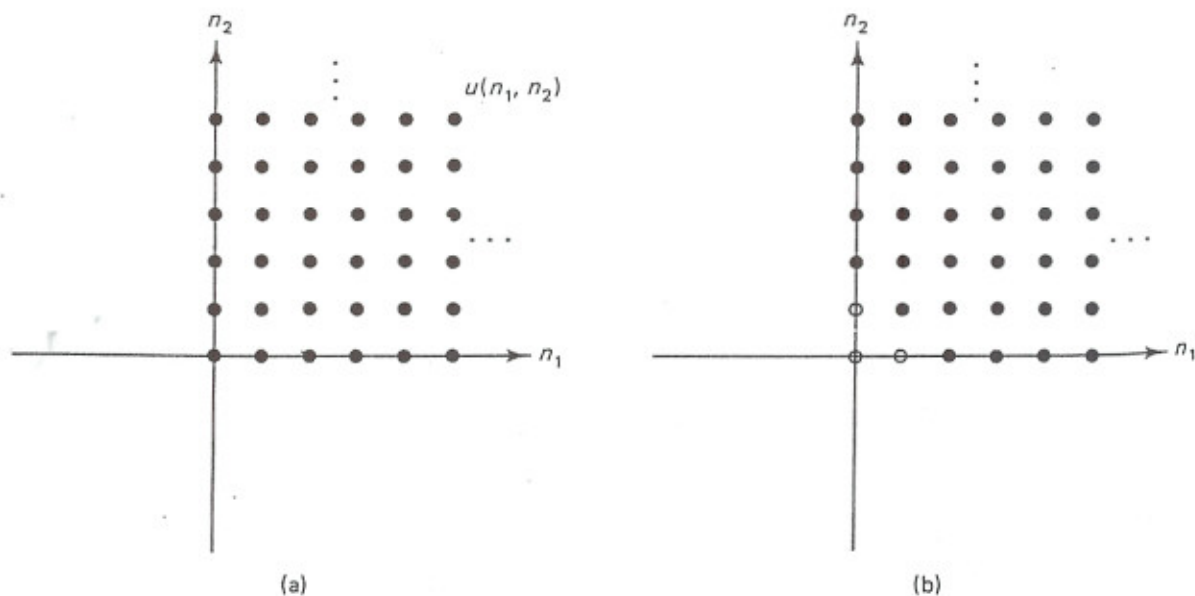
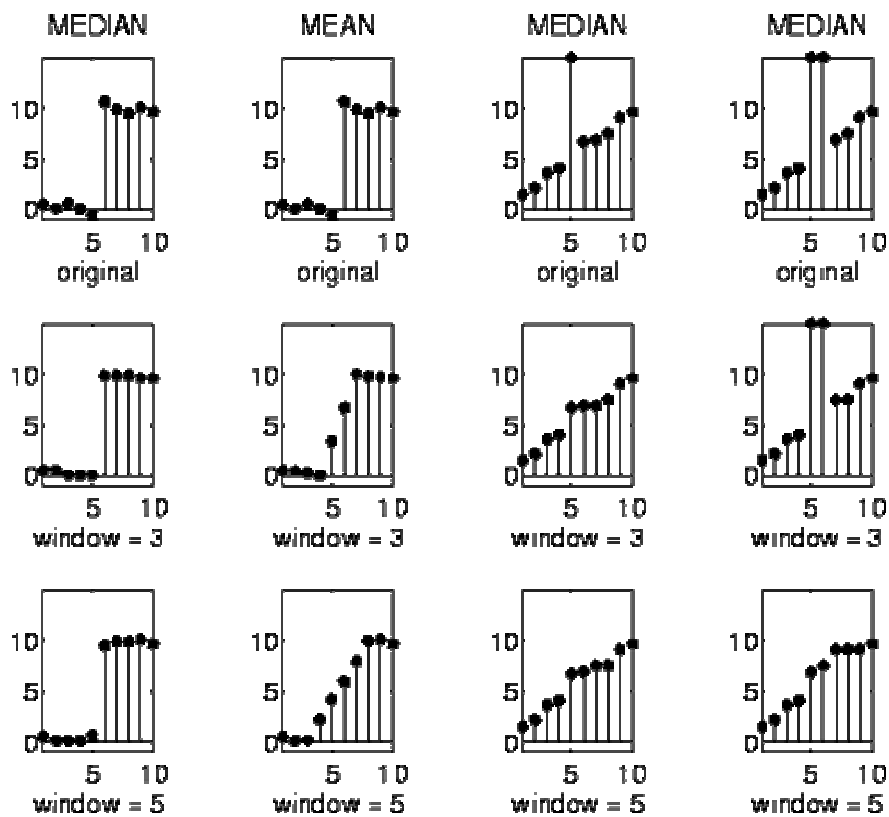


Figure 8.20 Illustration that a 2-D $N \times N$ -point median filter distorts 2-D step discontinuities. (a) Unit step sequence $u(n_1, n_2)$; (b) result of filtering $u(n_1, n_2)$ with a 5×5 -point median filter.

Median Filter

Median filtering is a non-linear signal enhancement technique for the smoothing of signals, the suppression of impulse [noise](#), and preserving of edges. In the one-dimensional case it consists of sliding a window of an odd number of elements along the signal, replacing the centre sample by the median of the samples in the window. In the following picture we use window sizes of 3 and 5 samples. The first two columns show a step function, degraded by some random noise. The two last columns show a noisy straight line, and in addition one and two samples, which are considerably different from the neighbour samples.



Whereas the median filter in the first column preserves the edge very well, the low-pass filtering method in the second column smooths the edge completely. Columns 3 and 4 show the importance of the window size: one sample out of range can be easily removed with a window size of 3, whereas two neighboring samples can only be removed with a larger window.

Median Filter

$$Y(n) = \text{med}(x_{n-k}, x_{n-k}, \dots, x_n, \dots, x_{n+k})$$

1. order x_j
2. choose middle element

More general

- 2b. choose i -th element

Allows removal of salt and not pepper or reverse

Recursive

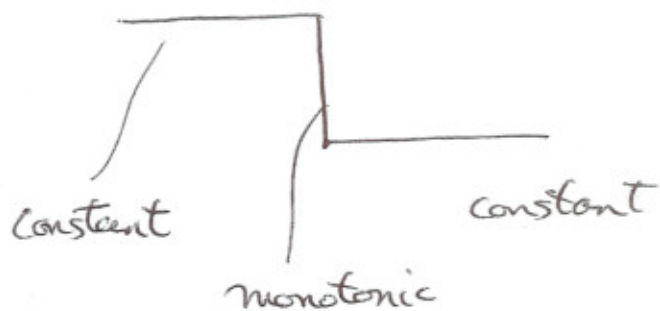
$$Y(n) = \text{med}(y_{n-k}, y_{n-k}, \dots, y_{n-1}, x_n, \dots, x_{n+k})$$

i.e. is done "in place" . It now depends on the order of the sweep!!

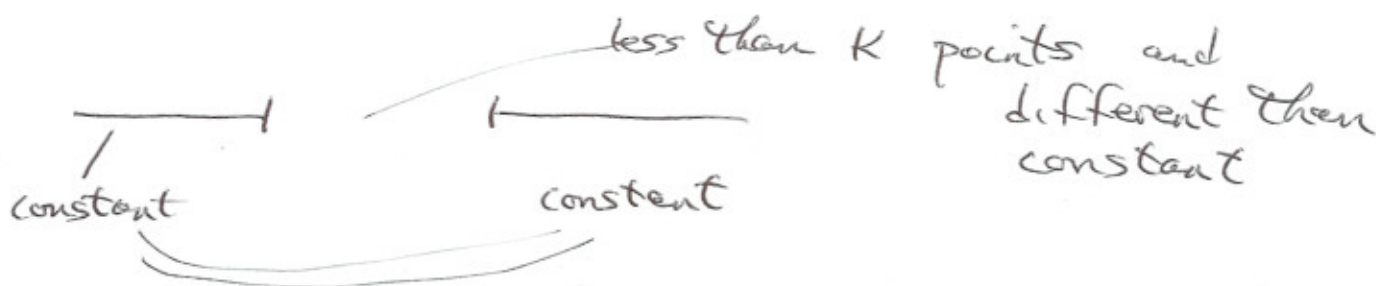
Gives more smoothing but more distortion

Def: Constant neighborhood
 $x_1 = x_2 = \dots = x_p \quad p \geq k+1$

Def: Edge



Def: Impulse



Def: Oscillation

not constant, edge or impulse

thm

A root signal consists only of constant neighborhoods and edges

thm

Repeated median \rightarrow root signal after finite number of iterations

For a filter window of size $2K+1$ and signal length L

typically 5-10 iterations is sufficient

Theorem

For a rank operator other than the median only the constant is a root signal

Definition:

white noise: independent identically distributed random variables with mean μ and variance σ^2

1. The median is the maximum likelihood estimate of the signal level in the presence of uncorrelated additive bi-exponential Gaussian distributed noise
2. The arithmetic average is the maximum likelihood estimate of the signal level in the presence of uncorrelated additive Gaussian distributed noise

Properties of Median

1. For odd number of points only pre-existing values can be chosen
2. For even number of points average of two middle values are chosen
3. Preserves discontinuity in one dimension (depends on window size).
4. It is a one dimensional operator and so does NOT preserve two dimensional discontinuity

Definition: An invariant subspace or root signal $x_n = \text{med}(x_{n-k}, x_{n-k}, \dots, x_n, \dots, x_{n+k})$

A constant neighborhood $x_1 = x_2 = \dots = x_p \quad p \geq K + 1$

An edge : Two constant neighborhoods with a discontinuity between them

An impulse: Two constant neighborhoods with K points between them

An oscillation: none of the above

Theorem: A root signal consists of only constant neighborhoods and edges

Definition: An impulse: Two constant neighborhoods with K points between them

Theorem: Repeated median yields a root signal after a finite number of iterations

For a filter window of size $2K+1$ and signal L the maximum number of iterations is $3 \frac{L-2}{2(K+2)}$.

Typically 5-10 is sufficient.

Theorem: For a rank operator other than the median only the constant is a root signal

Definition: white noise = independent identically distributed random variables with mean μ and variance σ^2

Theorem: The arithmetic average is the maximum likelihood estimate of the signal level in the presence of uncorrelated additive Gaussian distributed noise

Theorem: The median is the maximum likelihood estimate of the signal level in the presence of uncorrelated additive biexponential Gaussian distributed noise

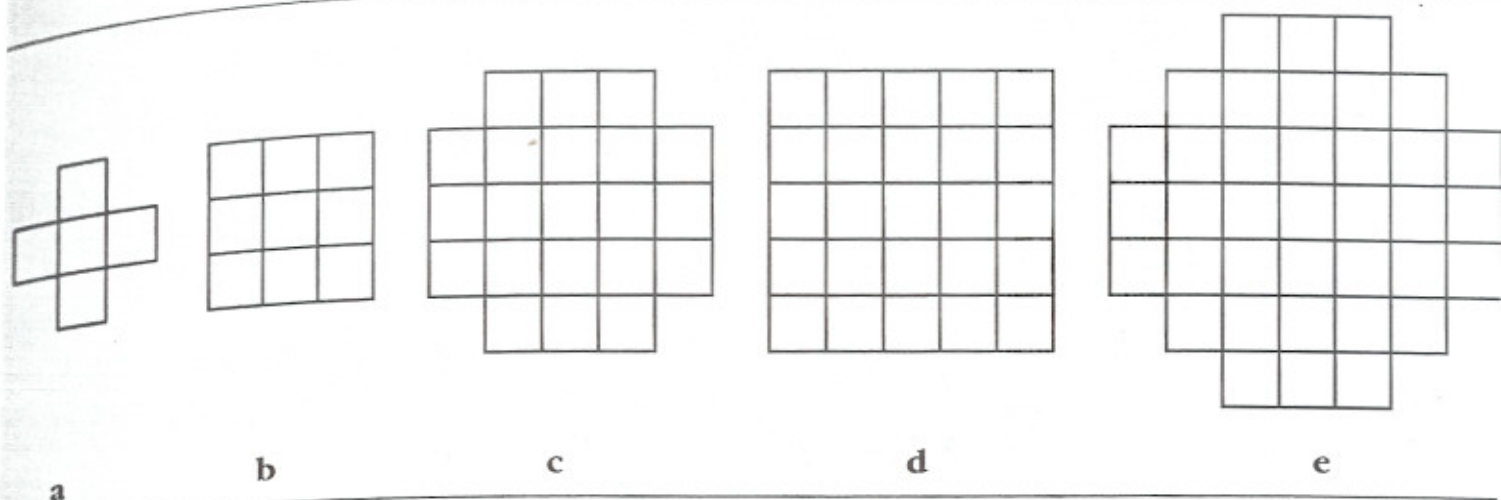


Figure 11. Neighborhood patterns used for median filtering:
a) 4 nearest-neighbor cross; **b)** 3×3 square containing nine pixels; **c)** 5×5 octagonal region with 21 pixels; **d)** 5×5 square containing 25 pixels; **e)** 7×7 octagonal region containing 37 pixels.

Neighborhood ranking

The use of weighting kernels to average together pixels in a neighborhood is a convolution operation, which has a direct counterpart in frequency space image processing. It is a linear operation in which no information is lost from the original image. There are other processing operations that can be performed in neighborhoods in the spatial domain that also provide noise smoothing. These are not linear and do not utilize or preserve all of the original data.

The most widely used of these methods is based on ranking of the pixels in a neighborhood according to brightness. Then, for example, the median value in this ordered list can be used as the brightness value for the central pixel. As in the case of the kernel operations, this is used to produce a new image and only the original pixel values are used in the ranking for the neighborhood around each pixel.

The so-called median filter is an excellent rejector of certain kinds of noise, for instance "shot" noise in which individual pixels are corrupted or missing from the image. If a pixel is accidentally changed to an extreme value, it will be eliminated from the image and replaced by a "reasonable" value, the median value in the neighborhood.

Figure 10 shows an example of this type of noise. Ten percent of the pixels in the original image, selected randomly, are set to black, and another ten percent to white. This is a rather extreme amount of noise. However, a median filter is able to remove the noise and replace the bad pixels with reasonable values while

apply it directly to your image data using `filter2`, or you can rotate it 180 degrees and use `conv2` or `convn`.

One simple filter `fspecial` can produce is an averaging filter. This type of filter computes the value of an output pixel by simply averaging the values of its neighboring pixels.

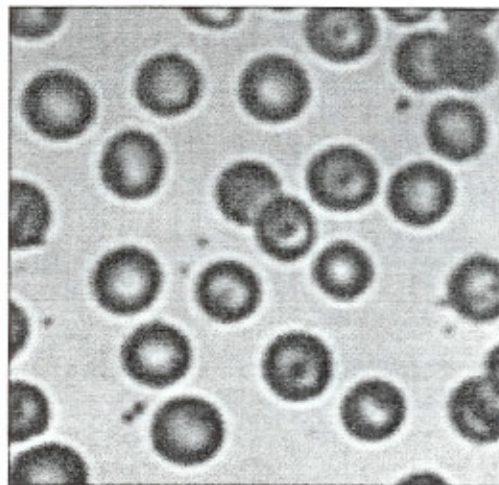
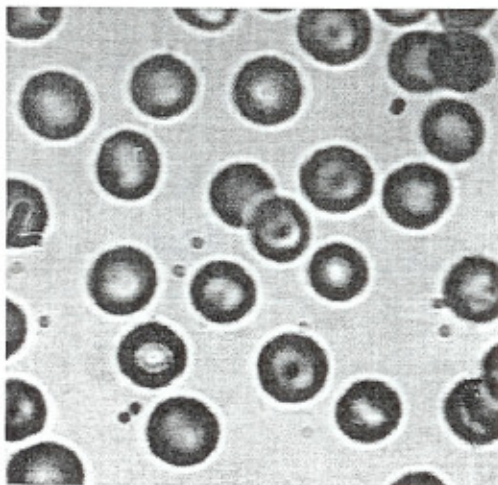
The default size of the averaging filter `fspecial` creates is 3-by-3, but you can specify a different size. The value of each element is $1/\text{length}(h(:))$. For example, a 5-by-5 averaging filter would be:

```
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
0.0400    0.0400    0.0400    0.0400    0.0400
```

Applying this filter to a pixel is equivalent to adding up the values of that pixel's 5-by-5 neighborhood and dividing by 25. This has the effect of smoothing out local highlights and blurring edges in an image.

This example illustrates applying a 5-by-5 averaging filter to an intensity image:

```
I = imread('blood1.tif');
h = fspecial('average',5);
I2 = uint8(round(filter2(h,I)));
imshow(I)
figure, imshow(I2)
```



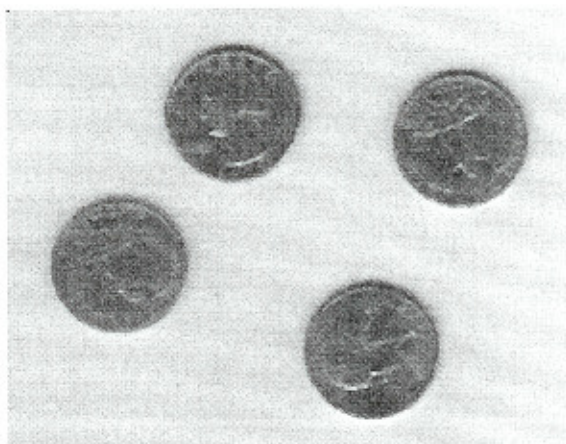
First, read in the image and add noise to it.

```
I = imread('eight.tif');  
J = imnoise(I,'salt & pepper',0.02);  
imshow(I)  
figure, imshow(J)
```



Now filter the noisy image and display the results. Notice that `medfilt2` does a better job of removing noise, with less blurring of edges.

```
K = filter2(fspecial('average',3),J)/255;  
L = medfilt2(J,[3 3]);  
imshow(K)  
figure, imshow(L)
```



Median filtering is a specific case of *order-statistic filtering*. For information about order-statistic filtering, see the reference entry for the `ordfilt2` function in Chapter 11.

Problem



Usually need noise reduction before performing higher level processing steps.

Spatial Filtering

- 1) Define a center point (x, y)
- 2) Perform an operation that involves only pixels in a predefined neighborhood about that center point
- 3) Let the result be the response of the process at the point
- 4) Repeat the process for every point in the image

*If operations performed is linear,
it is called linear filtering;
otherwise it is called nonlinear filtering.*

*We are interested in filtering operations that are performed directly on the pixels on an image. Frequency domain filtering may be covered by the professor.

Linear Filters: Weighted Averaging

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

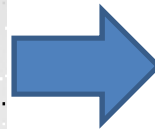
$g(x,y)$

$$g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l)$$

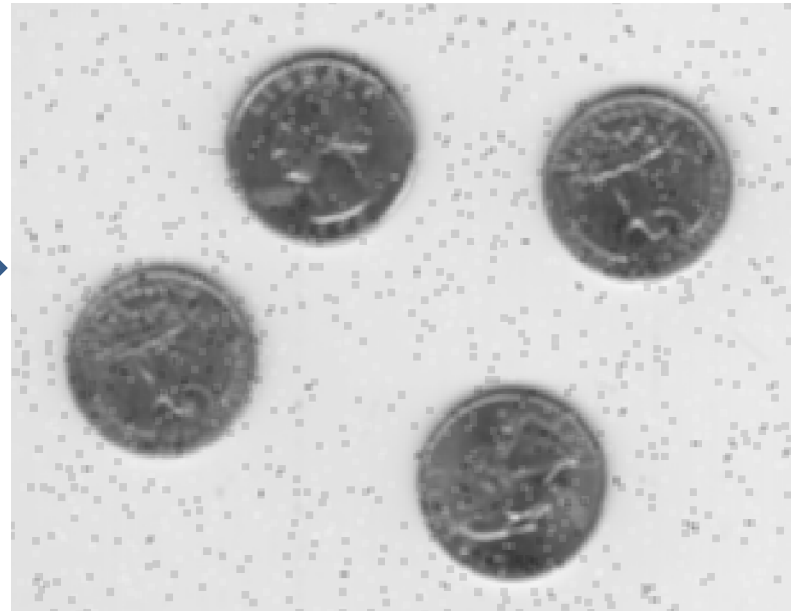
Linear Filters: Mean Filter



Input image



Mean
Filter



Output image

When a mean filter is applied, it removes noise a little.
However, it gives BLURRY result.

Nonlinear Filters: Median Filter

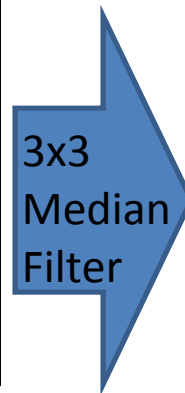
- A non-linear combination of neighboring pixel may give better results.
- Median Filter
 - Similar to mean filter. It has a moving mask. Instead of taking mean, it takes median.
 - It cannot be done with a convolution, instead we need to sort all the pixels under mask, take the value in the center*.

*For a more efficient algorithm, go to http://en.wikipedia.org/wiki/Median_search)

Nonlinear Filters: Median Filter

144	117	214	140	147
48	251	101	195	7
109	18	30	189	2
53	192	4	107	120
100	86	85	114	27

Input



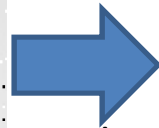
0	101	117	101	0
48	109	140	140	7
48	53	107	101	7
53	85	86	85	27
0	53	85	27	0

Output

Nonlinear Filters: Median Filter



Input image



Median
Filter



Output image

- Median filter is usually good to remove outlier pixels from an image.
- Such filters preserve edge information while removing noise.
- However, median filter reduces the quality of the image.

(c) פילטרים לא לינאריים
התמונה לאחר רעש גאוסייני עברה פילטר median עם מסכה 3X3. התהליך בוצע 10 פעמים. להלן תוצאות הסינון וכן ההפרשים בין השלבים השונים (ההפרשים מוצגים באמצעות imagesc לכן יש להתייחס בעיקר לכמות השינויים).

noised - gaussian



1 median filters



2 median filters



3 median filters



4 median filters



5 median filters



6 median filters



7 median filters



8 median filters

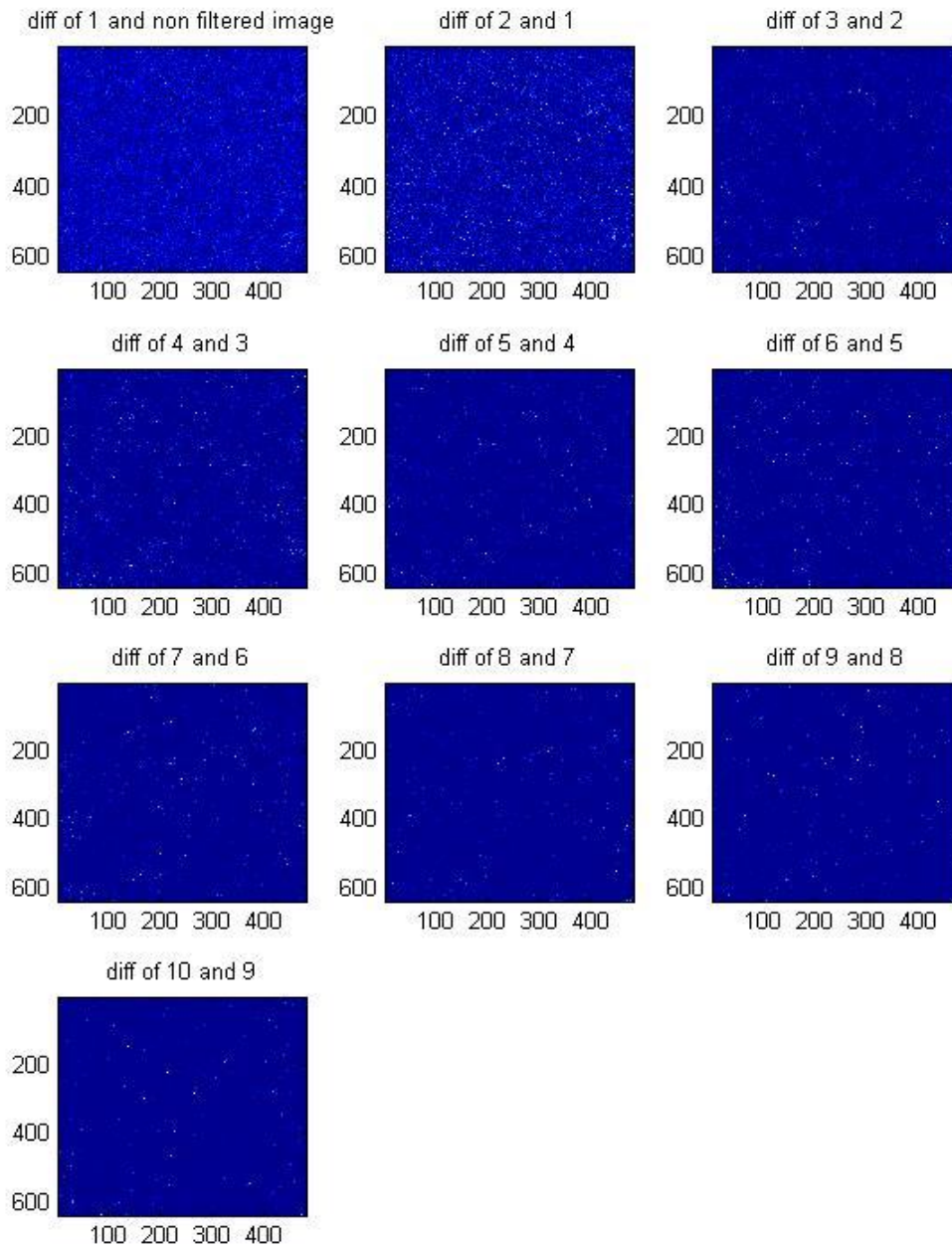


9 median filters



10 median filters





ניתן לראות שעיקר פעולת הסינון התבצעה בהפעלה הראשונה של הפילטר וקצת בשניה. בשאר הפעמים הפעלטר בעיקר טשטש את התמונה (אפשר לראות את הטשטוש על השיער של הילדה הגדולה). עם זאת... גם לאחר 10 איטרציות התמונה נשארה די ברורה כשאזורים יחסית קבועים אמנם מטושטשים מעט, אבל מעברים בין חלקים שונים בתמונה (למשל מכנסיים ורקע) נותרו חדים למדי. בתמונות ההפרשים רואים שההליך פחות או יותר מתכנס.

התמונה לאחר רעש salt & pepper עברה פילטר median עם מסכה 3X3. התהליך בוצע 10 פעמים. להלן תוצאות הסינון וכן ההפרשים בין השלבים השונים (ההפרשים מוצגים באמצעות imagesc לכן יש להתייחס בעיקר לכמות השינויים).

noised - S&P



1 median filters



2 median filters



3 median filters



4 median filters



5 median filters



6 median filters



7 median filters



8 median filters

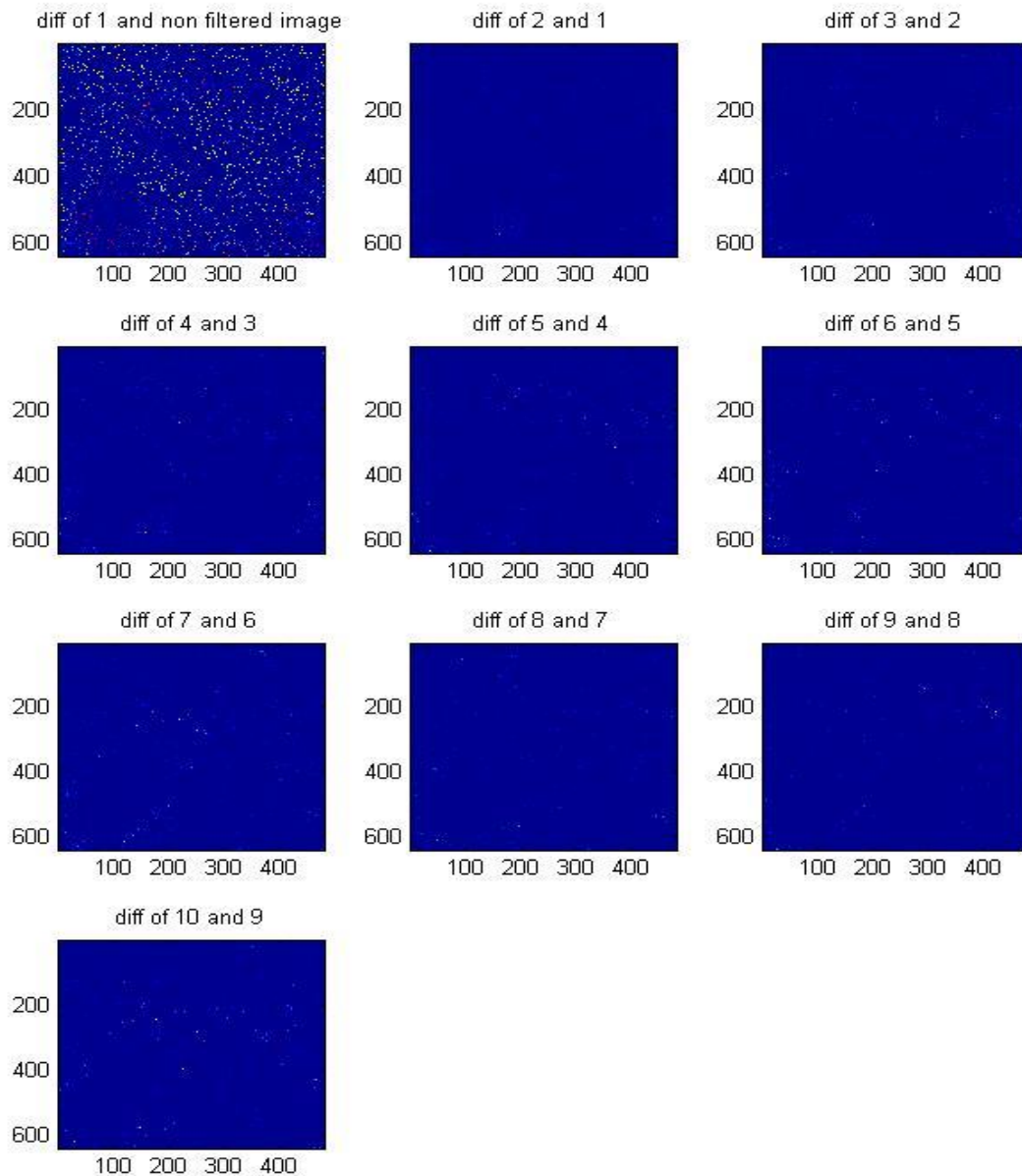


9 median filters



10 median filters





עבור רעש S&P פעולת הסינון כולה התבצעה בהפעלה הראשונה של הפילטר. בכל שאר האיטרציות קיבלנו בעיקר טשטוש של התמונה. ושוב.. גם לאחר 10 איטרציות התמונה נשארה די ברורה גם לאחר כל הסינונים. בתמונות ההפרשים ברור מאוד רואים שהרעש כולו מסונן בפעם הראשונה ושההליך פחות או יותר מתכנס.



(a) Image with binary noise



(b) 3×3 median filtered



(c) image with Gaussian noise



(d) 3×3 median filtered.

Figure 7.21 Median filtering.



(a)



(b)



(c)



(d)

Figure 4.23 (a) Original image; (b) image corrupted by impulse noise; (c) result of 5×5 neighborhood averaging; (d) result of 5×5 median filtering. (Courtesy of Martin Connors, Texas Instruments, Inc., Lewisville, Tex.)

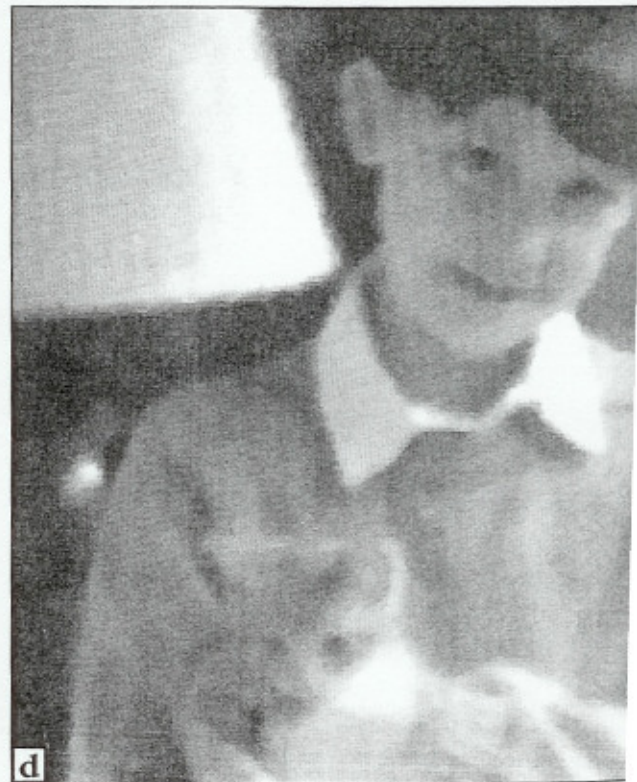


Figure 10. Removal of shot noise with a median filter:

- a) original image;*
- b) image a with 10% of the pixels randomly selected and set to black, and another 10% randomly and set to white;*
- c) application of median filtering to image b using a 3×3 square region;*
- d) application of median filtering to image b using a 5×5 octagonal region.*



Figure 13. Repeated application of a 5×5 octagonal median filter:

a) original image;

b) after 12 applications. The fine details have been erased and textured regions leveled to a uniform shade of grey, but boundaries have not shifted.



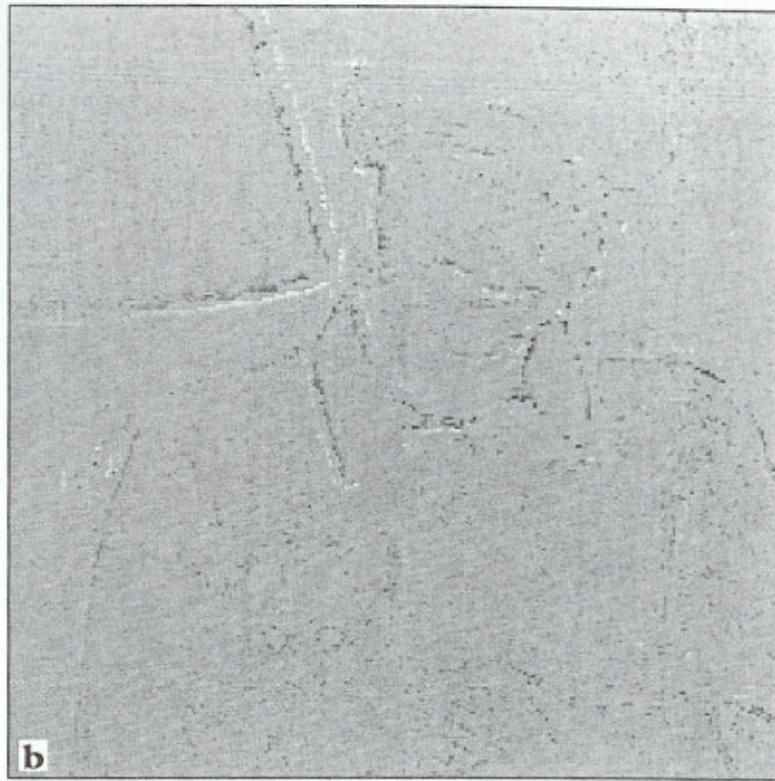


Figure 15. Application of the truncated median filter to posterize the image from Figure 10:

- a)** one application of the 3×3 truncated median;
- b)** difference between Figure **a** and a conventional 3×3 median filter, showing the difference in values along edges;
- c)** 12 applications of the truncated median filter.

Algorithm 4.3: Efficient median filtering

1. Set

$$th = \frac{mn}{2}$$

2. Position the window at the beginning of a new row, and sort its contents. Construct a histogram H of the window pixels, determine the median med , and record lt_med , the number of pixels with intensity less than or equal to med .

3. For each pixel p in the leftmost column of intensity p_g , perform

$$H[p_g] = H[p_g] - 1$$

Further, if $p_g < med$, set

$$lt_med = lt_med - 1$$

4. Move the window one column right. For each pixel p in the rightmost column of intensity p_g , perform

$$H[p_g] = H[p_g] + 1$$

If $p_g < med$, set

$$lt_med = lt_med + 1$$

5. If $lt_med > th$ then go to 6.

Repeat

$$lt_med = lt_med + H[med]$$

$$med = med + 1$$

until $lt_med \geq th$. Go to 7.

6. Repeat

$$med = med - 1$$

$$lt_med = lt_med - H[med] \tag{4.35}$$

until $lt_med \leq th$.

7. If the right-hand column of the window is not at the right-hand edge of the image, go

8. If the bottom row of the window is not at the bottom of the image, go to step 2.

Effect of median filtering is shown in Figure 4.14.

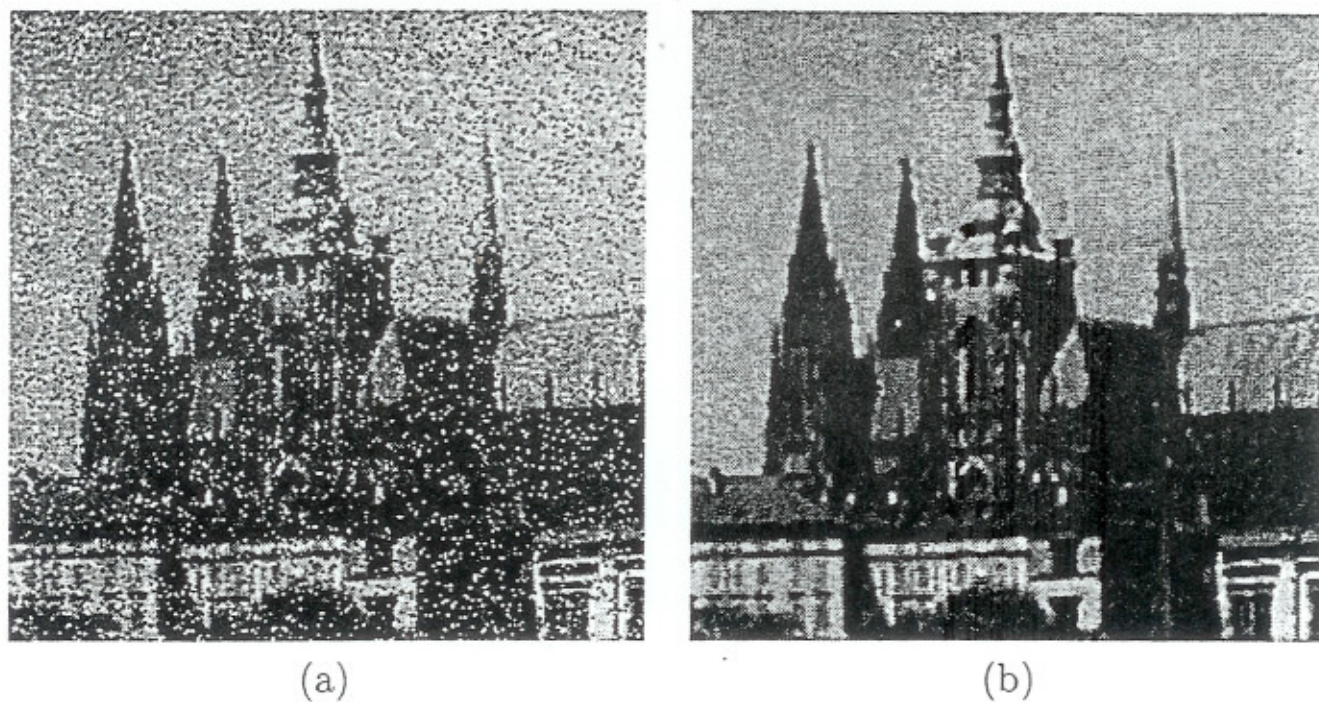


Figure 4.14: Median filtering: (a) image corrupted with impulse noise (14% of image are red with bright and dark dots); (b) result of 3×3 median filtering.

The main disadvantage of median filtering in a rectangular neighborhood is its damaging effect on lines and sharp corners in the image—this can be avoided if another shape of neighborhood is used. For instance, if horizontal/vertical lines need preserving, a neighborhood such as that in Figure 4.15 can be used.

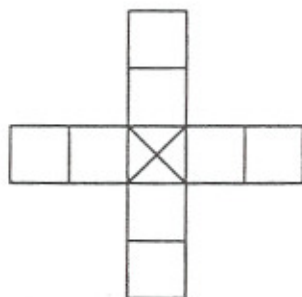


Figure 4.15: Horizontal/vertical line preserving neighborhood for median filtering.

Median smoothing is a special instance of more general rank filtering techniques [Rose and Kak 82, Yaroslavskii 87], the idea of which is to order pixels in some neighborhood sequence. The results of pre-processing are some statistics over this sequence, of which the median is one possibility. Another variant is the maximum or the minimum values over the sequence. This defines generalizations of dilation and erosion operators (Chapter 11) that deal with more brightness values.

A similar generalization of median techniques is given in [Borik et al. 83]. Their method is called order statistics (OS) filtering. Values in the neighborhood are again ordered in

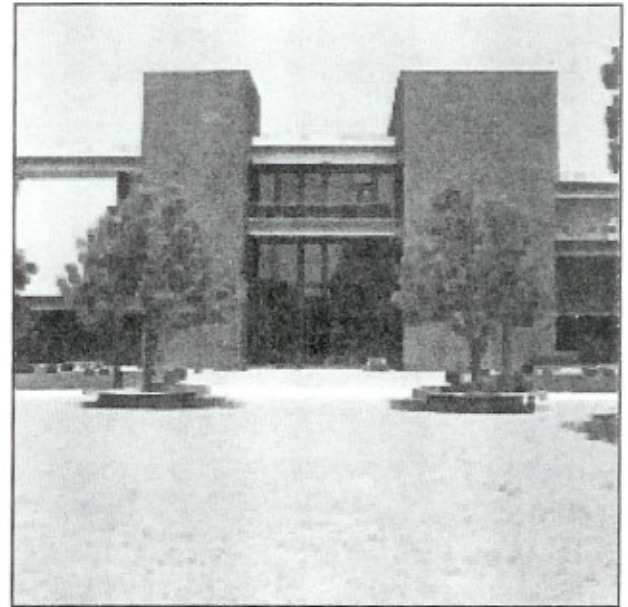
to the fact that the images are much larger than the masks, and these “wasted” rows and columns are often filled with zeros (or cropped off the image). For example, with a 3×3 mask, we lose one outer row and column, a 5×5 loses two rows and columns—this is not usually significant for a typical 256×256 or 512×512 image.

The maximum and minimum filters are two order filters that can be used for elimination of salt-and-pepper (impulse) noise. The *maximum filter* selects the largest value within an ordered window of pixel values, whereas the *minimum filter* selects the smallest value. The minimum filter works when the noise is primarily of the salt-type (high values), and the maximum filter works best for pepper-type noise (low values). In Figures 3.3-2a, b, the application of a minimum filter to an image contaminated

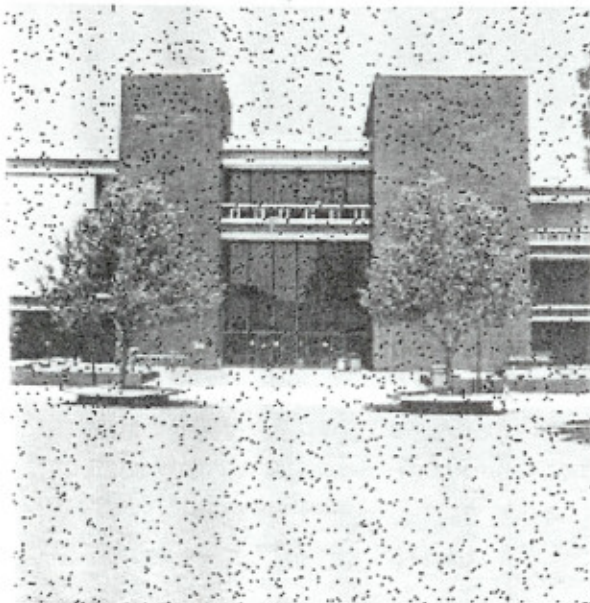
Figure 3.3-2 Minimum and Maximum Filters



a. Image with salt noise; probability of salt = .04.



b. Result of minimum filtering image (a); mask size = 3×3 .



c. Image with pepper noise; probability of pepper = .04.

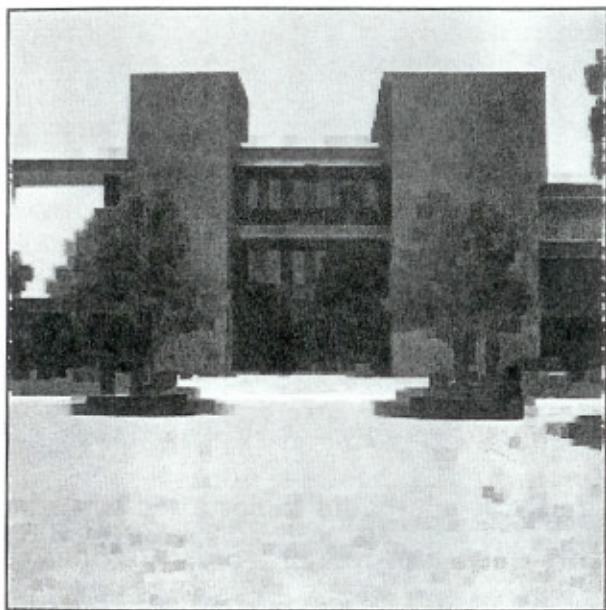


d. Result of maximum filtering image (c); mask size = 3×3 .

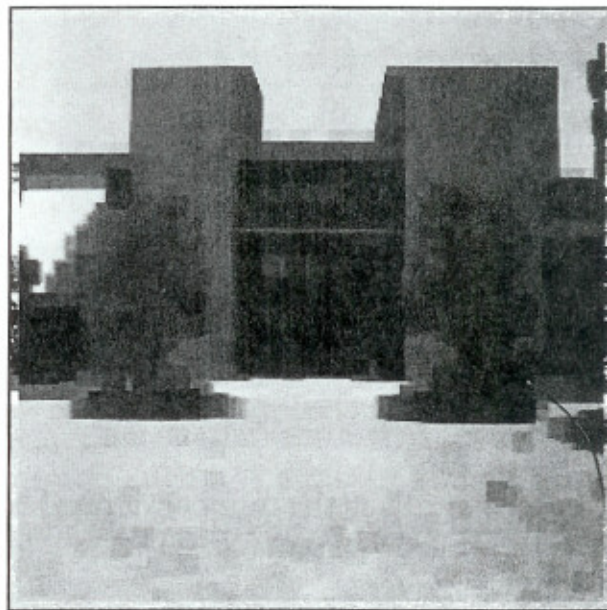
with salt-type noise is shown, and in Figures 3.3-2c, d a maximum filter applied to an image corrupted with pepper-noise is shown. Here we see that these filters are excellent for this type of noise, with minimal information loss. As the size of the window gets bigger, the more information loss occurs; with windows larger than about 5×5 the image acquires an artificial, "painted," effect (Figure 3.3-3).

In a manner similar to the median, minimum, and maximum filter, order filter can be defined to select a specific pixel rank within the ordered set. For example, we may find for certain types of pepper noise that selecting the second highest value works better than selecting the maximum value. This type of ordered selection is very

Figure 3.3-3 Various Window Sizes for Maximum and Minimum Filters



a. Result of minimum filtering Figure 3.3-2a; mask size = 5×5 .



b. Result of minimum filtering Figure 3.3-2a; mask size = 9×9 .



c. Result of maximum filtering Figure 3.3-2c; mask size = 5×5 .



d. Result of maximum filtering Figure 3.3-2c; mask size = 9×9 .

The final two order filters are the midpoint and alpha-trimmed mean filter. They are actually both order and mean filters because they rely on ordering the pixel values, but they are then calculated by an averaging process. The *midpoint filter* is the average of the maximum and minimum within the window, as follows:

$$\text{ordered set} \rightarrow I_1 \leq I_2 \leq I_3 \leq \dots \leq I_{N^2}$$

$$\text{Midpoint} = \frac{I_1 + I_{N^2}}{2}$$

The midpoint filter is most useful for gaussian and uniform noise, as illustrated in Figure 3.3-4.

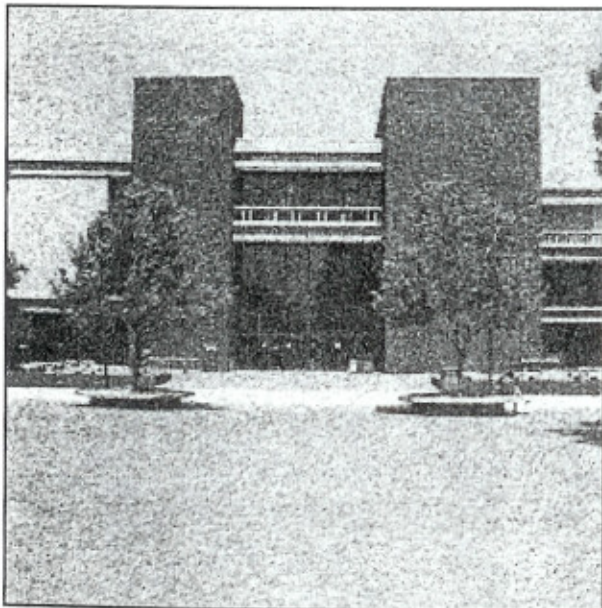
The *alpha-trimmed mean* is the average of the pixel values within the window but with some of the endpoint-ranked values excluded. It is defined as follows:

$$\text{ordered set} \rightarrow I_1 \leq I_2 \leq \dots \leq I_{N^2}$$

$$\text{Alpha-trimmed mean} = \frac{1}{N^2 - 2T} \sum_{i=T+1}^{N^2-T} I_i$$

where T is the number of pixel values excluded at each end of the ordered set, and can range from 0 to $(N^2 - 1)$.

Figure 3.3-4 Midpoint Filter



a. Image with gaussian noise—variance = 300; mean = 0.

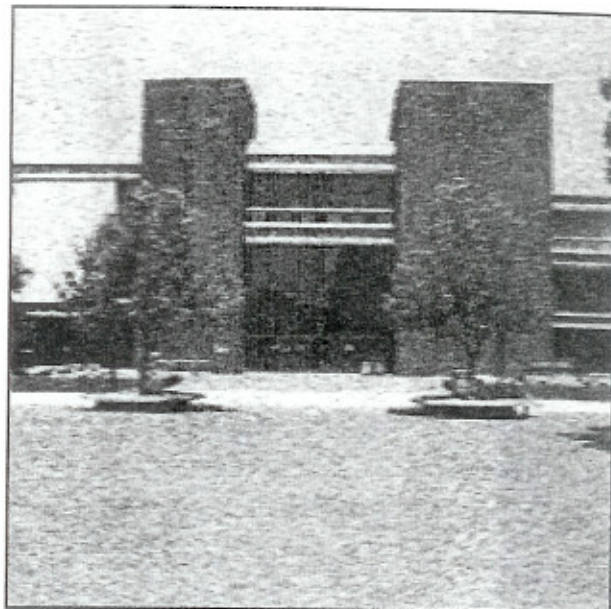


b. Result of midpoint filter; mask size = 3.

Figure 3.3-4 (Continued)



c. Image with uniform noise—variance = 300;
mean = 0.



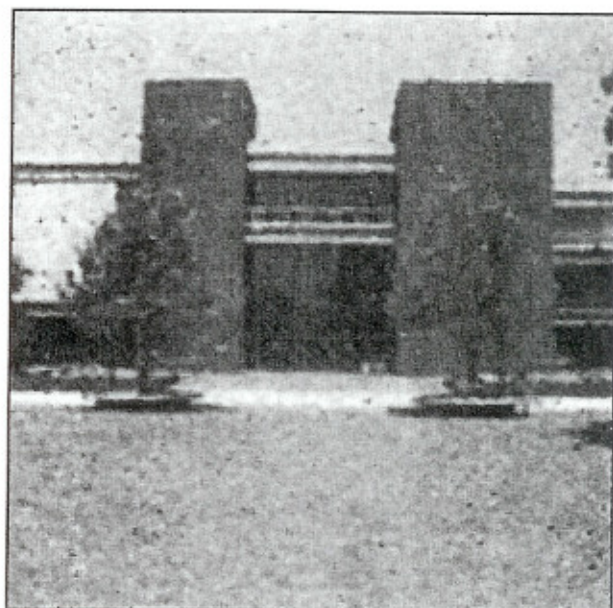
d. Result of midpoint filter; mask size = 3.

The alpha-trimmed mean filter ranges from a mean to median filter, depending on the value selected for the T parameter. For example, if $T = 0$, the equation reduces to finding the average gray-level value in the window, which is an arithmetic mean filter. If $T = (N^2 - 1)/2$, the equation becomes a median filter. This filter is useful for images containing multiple types of noise, such as gaussian and salt-and-pepper noise. In Figure 3.3-5 are the results of applying this filter to an image with both gaussian and salt-and-pepper noise.

Figure 3.3-5 Alpha-Trimmed Mean Filter

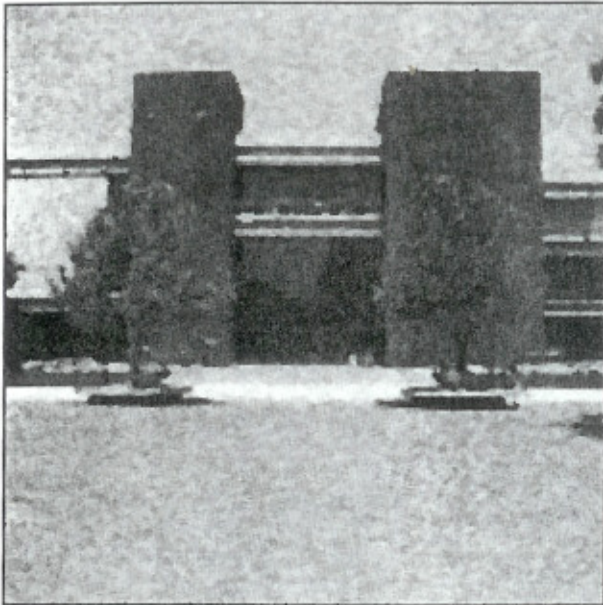


a. Image with gaussian and salt-and-pepper
noises—variance = 200; mean = 0.

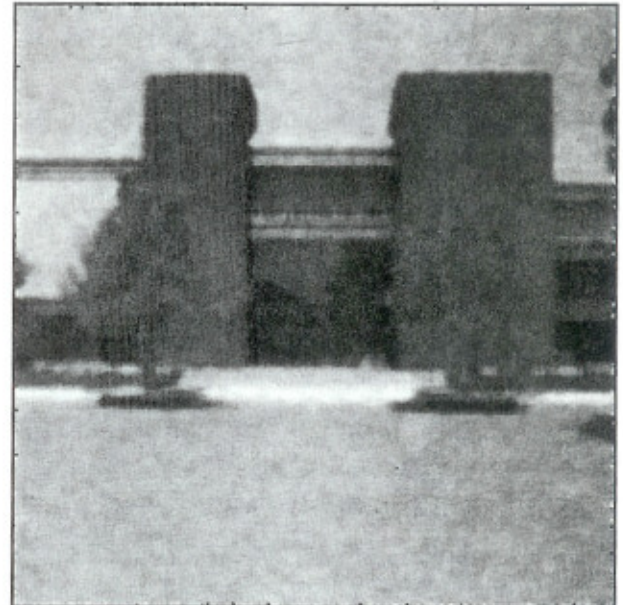


b. Result of alpha-trimmed mean filter; mask size =
3, trim size = 1.

Figure 3.3-5 (Continued)



c. Result of alpha-trimmed mean filter; mask size = 3, trim size = 4.



d. Result of alpha-trimmed mean filter; mask size = 5, trim size = 5.

3.3.2 Mean Filters

The mean filters function by finding some form of an average within the $N \times N$ window, using the sliding window concept to process the entire image. The most basic of these filters is the *arithmetic mean filter*, which finds the arithmetic average of the pixel values in the window, as follows:

Figure 3.3-6 Arithmetic Mean Filter



a. Image with gaussian noise—variance = 300; mean = 0.









b. Image with gamma noise—variance = 300; alpha = 1.

פילטר α -trim

יתרוננו של פילטר זה הוא ביכולתו לסנן רעשים קיצוניים ולמצע את שאר האות. לכן, על פניו הוא יכול להתמודד גם עם רעש גאוסיאני וגם עם רעש מלח-פלפל.

נבחר חלון בגודל $7*7$ על מנת להתגבר בצורה טובה על הרעש הגאוסיאני. בחלון בגודל זה, נכנסים מספר פיקסלי מלח ופלפל שנדרש "לקצץ" (trim). בחירה בקיצוץ של 3 ערכים קיצוניים בכל חלון מניבה תוצאות טובות. זאת לעומת קיצוץ של ערך יחיד, אשר משאיר את רעש המלח והפלפל (אם כי מושטש).

<p>חלון $7*7$ קיצוץ 3 ערכי קיצוץ – רעש גאוסיאני</p> <p>filtered (gaussian noise)</p>	<p>חלון $7*7$ קיצוץ ערך קיצוץ 1 – מלח פלפל</p> <p>filtered (salt & pepper)</p>	<p>חלון $7*7$ קיצוץ 3 ערכי קיצוץ – מלח פלפל</p> <p>filtered (salt & pepper)</p>
		
<p>[original pic - filtered pic] (gaussian noise). L2 Norm=39.7924</p>	<p>[original pic - filtered pic] (salt & pepper). L2 Norm=40.1501</p>	<p>[original pic - filtered pic] (salt & pepper). L2 Norm=38.8263</p>
		

כיוון שהמוצע הינו לא ממושקל, ניתן לראות כי ההתגברות על הרעש הגאוסיאני אינה טובה, ונוצר רעש נוסף ו-artifacts מהמיצוע של מספר רעשים סמוכים כמו בהגדלה להלן, הלקוחה מאזור חלק בתמונה המורעשת:



גם בפילטר של התמונה המקורית ניתן לראות artifacts כאלה, לדוגמה, בין שני הפסים על מכסה המנוע:

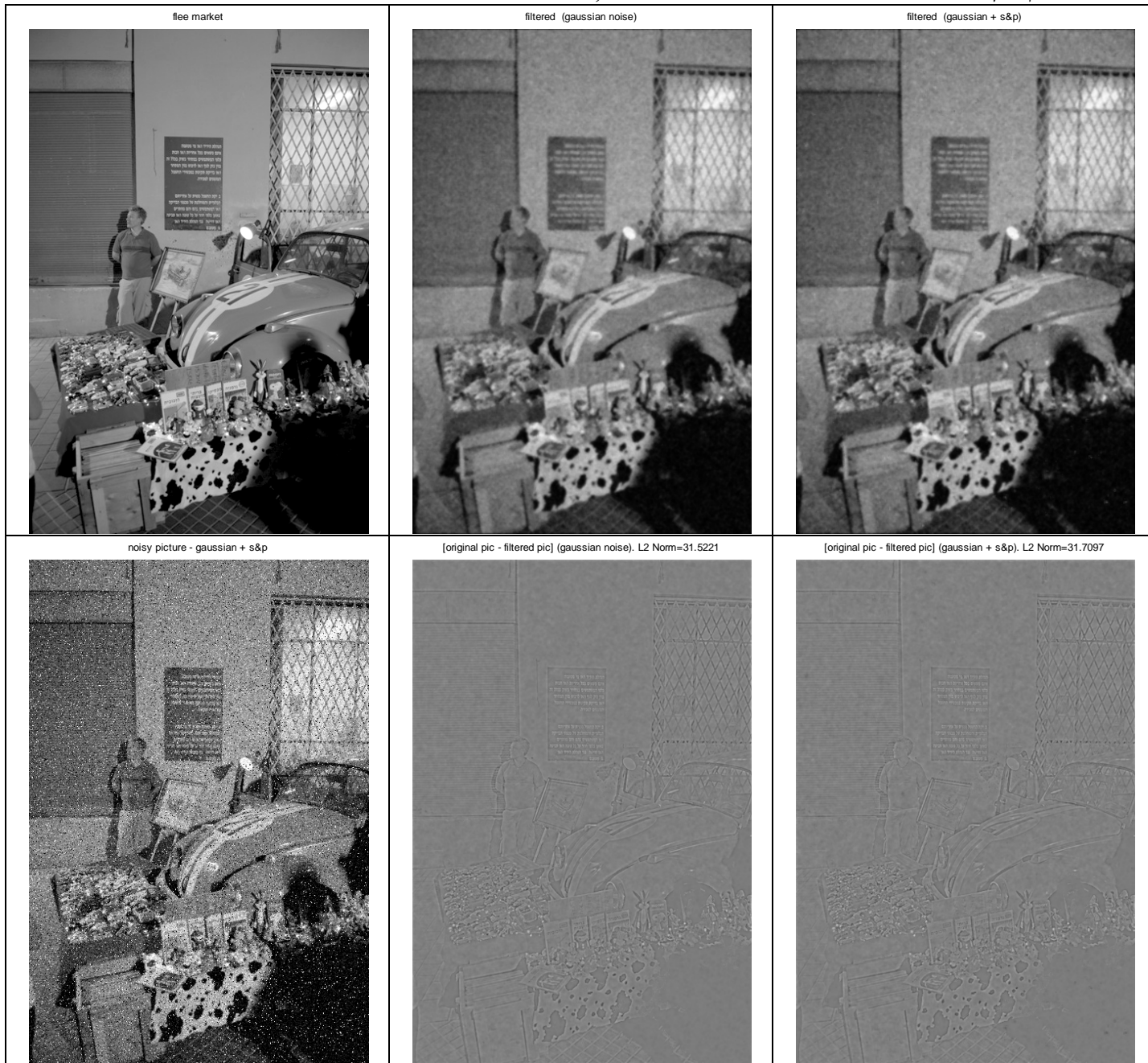


1	6	15	20	15	6	1
6	36	90	120	90	36	6
15	90	225	300	225	90	15
20	120	300	400	300	120	20
15	90	225	300	225	90	15
6	36	90	120	90	36	6
1	6	15	20	15	6	1

על מנת להתגבר על בעיה זו נפעיל את המיצוע באופן ממושקל. המשקלות יהיו (עבור חלון 7×7): /4096
ניתן לראות כי עבור התמונה הלא מורעשת, התוצאה השתפרה והוסרו ה-artifacts:



בתמונות המורעשות, ניתן לראות כי השימוש במשקולות הקטין את השגיאה עבור רעש גאוסיאני (39.8 ללא משקולות בעמוד הקודם, 31.5 עם משקולות כאן). ניתן לראות גם כי הפילטר מצליח להתמודד היטב עם תמונה הכוללת רעש גאוסיאני ומלח-פלפל. למעשה, השימוש ב-"קיצוץ" מצליח להביא את התמונה עם הרעש המשולב, לאותם התוצאות כמו עבור רעש גאוסיאני בלבד.



התמונה לאחר רעש salt & pepper עברה פילטר alpha-trim עם מסכה 3X3 ו- $\alpha=0.2$. התהליך בוצע 10 פעמים. להלן תוצאות הסינון וכן ההפרשים בין השלבים השונים (ההפרשים מוצגים באמצעות imagesc לכן יש להתייחס בעיקר לכמות השינויים).

noised - S&P



1 alpha trim filters



2 alpha trim filters



3 alpha trim filters



4 alpha trim filters



5 alpha trim filters



6 alpha trim filters



7 alpha trim filters



8 alpha trim filters

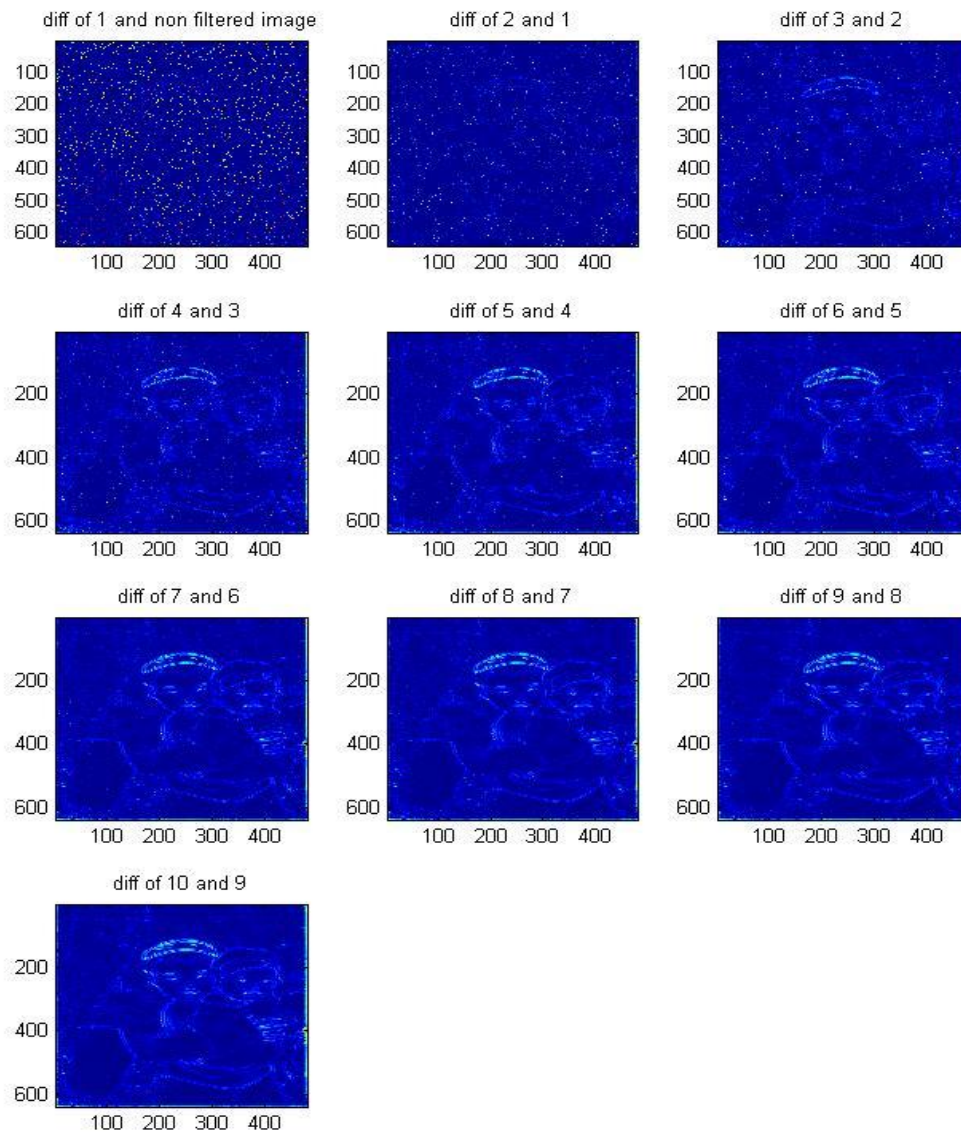


9 alpha trim filters



10 alpha trim filters





פעולת הסינון התבצעה בעיקר בשתי האיטרציות הראשונות של הפילטר – אחר כך קיבלנו בעיקר טשטוש. alpha-trim מטפל לא רע ברעש S&P, אבל עדיין פחות טוב מחציון. כאן התהליך לא התכנס, בגלל פעולת המיצוע של הפילטר שתוצאותיה ניכרות בקצוות שבתמונה.

התמונה לאחר רעש גאוסייני עברה פילטר α -trim עם מסכה 3×3 ו- $\alpha=0.2$. התהליך בוצע 10 פעמים. להלן תוצאות הסינון וכן הפרשים בין השלבים השונים (ההפרשים מוצגים באמצעות imagesc לכן יש להתייחס בעיקר לכמות השינויים).

noised - gaussian



1 alpha trim filters



2 alpha trim filters



3 alpha trim filters



4 alpha trim filters



5 alpha trim filters



6 alpha trim filters



7 alpha trim filters



8 alpha trim filters

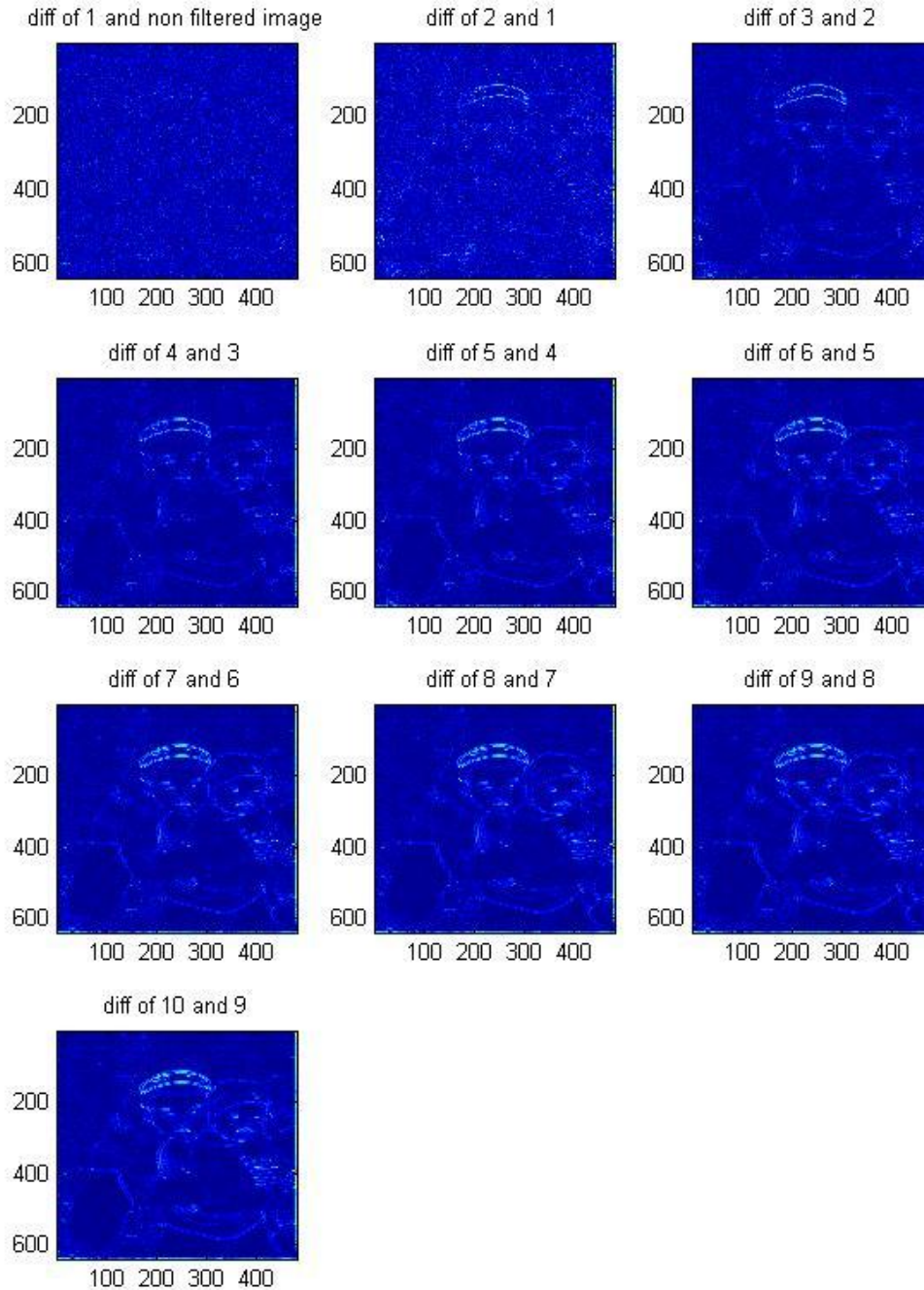


9 alpha trim filters



10 alpha trim filters





פעולת הסינון התבצעה בעיקר בשתי האיטרציות הראשונות של הפילטר – אחר כך קיבלנו בעיקר טשטוש. α -trim מטפל ברעש גאוסייני מעט טוב יותר מהחציון. כאן התהליך לא התכנס, בגלל פעולת המיצוע של הפילטר שתוצאותיה ניכרות בקצוות שבתמונה. מכיוון שהרעש הגאוסייני מייצר פיקסלים "פחות בולטים" ניתן לראות את אפקט הטשטוש של הקצוות כבר בהפעלה השנייה של הפילטר.

applied to images with various types of noise. It can be seen that the larger the mask size, the more pronounced the blurring effect. This type of filter works best with gaussian and uniform noise. The blurring effect, which reduces image details, is undesirable, and the other mean filters are designed to minimize this loss of detail information.

The *contra-harmonic mean filter* works well for images containing salt OR pepper type noise, depending on the filter order R :

Figure 3.3-7 Contra-Harmonic Mean Filter



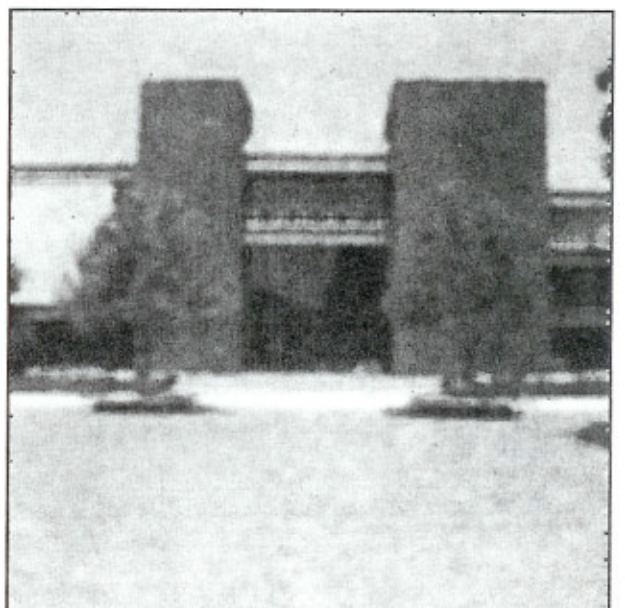
a. Image with salt noise—probability = .04.



b. Result of contra-harmonic mean filter; mask size = 3; order = -3.



c. Image with pepper noise—probability = .04.



d. Result of contra-harmonic mean filter; mask size = 3; order = +3.

$$\text{Contra-Harmonic Mean} = \frac{\sum_{(r,c) \in W} d(r,c)^{R+1}}{\sum_{(r,c) \in W} d(r,c)^R}$$

where W is the $N \times N$ window under consideration.

For negative values of R , it eliminates salt-type noise, whereas for positive values of R , it eliminates pepper-type noise. This is shown in Figure 3.3-7.

The *geometric mean filter* works best with gaussian noise and retains detail information better than an arithmetic mean filter. It is defined as the product of pixel values within the window, raised to the $1/N^2$ power:

$$\text{Geometric Mean} = \prod_{(r,c) \in W} [I(r,c)]^{\frac{1}{N^2}}$$

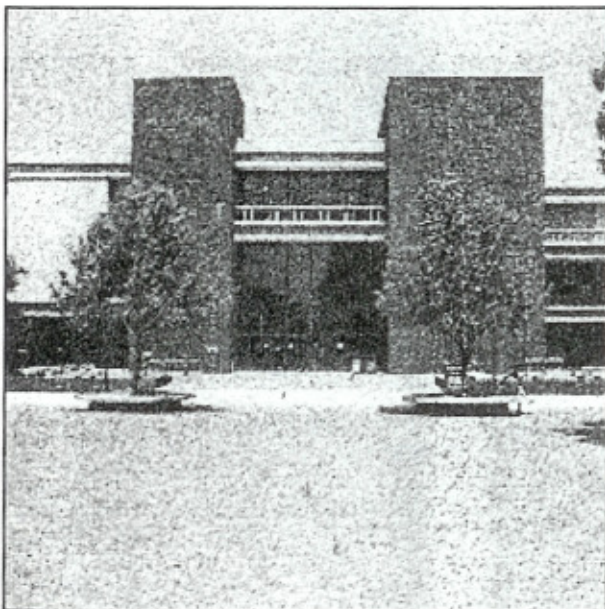
In Figure 3.3-8 are the results of applying this filter to images with gaussian (a , b) and pepper noise (c , d). As shown in Figure 3.3-8d, this filter is ineffective in the presence of pepper noise—with zero (or very low) values present in the window, the equation returns a zero (or very small) number.

The *harmonic mean filter* also fails with pepper noise but works well for salt noise. It is defined as follows:

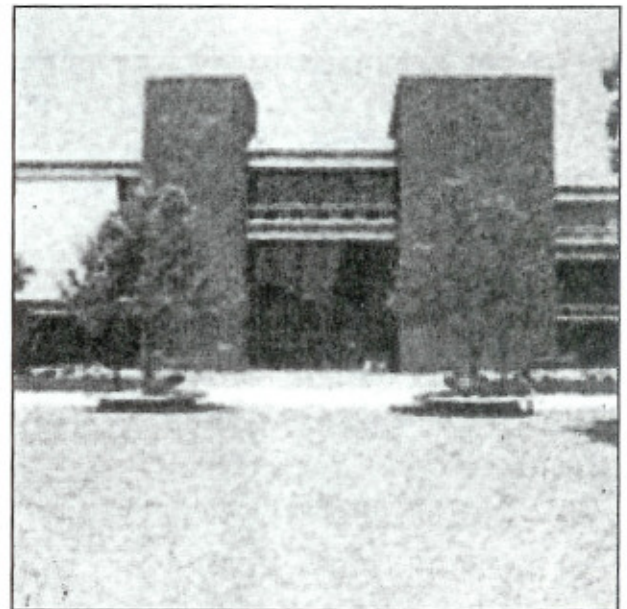
$$\text{Harmonic Mean} = \frac{N^2}{\sum_{(r,c) \in W} \frac{1}{d(r,c)}}$$

This filter also works with gaussian noise, retaining detail information better than the arithmetic mean filter. In Figure 3.3-9 are the results from applying the harmonic

Figure 3.3-8 Geometric Mean Filter

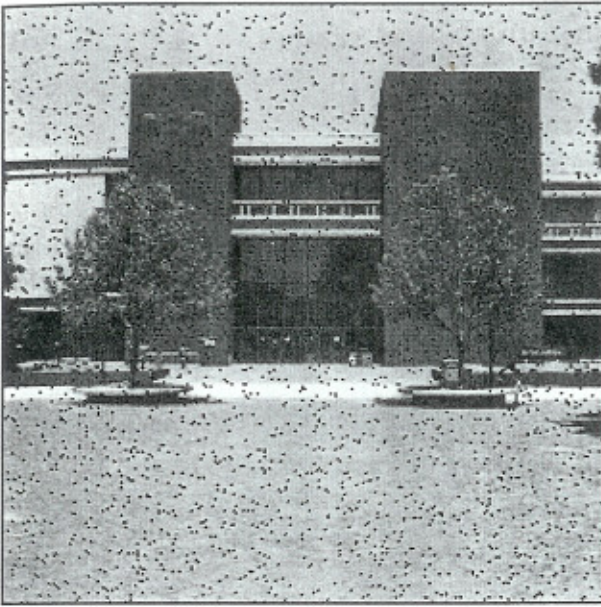


a. Image with gaussian noise—variance = 300; mean = 0.

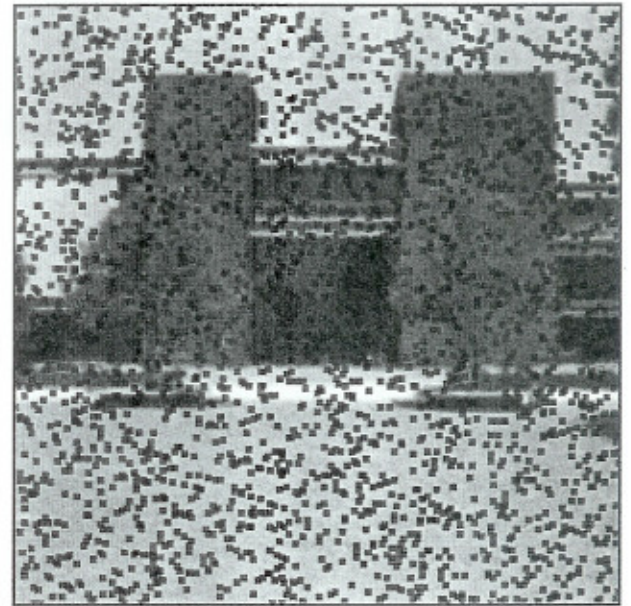


b. Result of geometric mean filter on image with gaussian noise; mask size = 3.

Figure 3.3-8 (Continued)



c. Image with pepper noise—probability = .04.



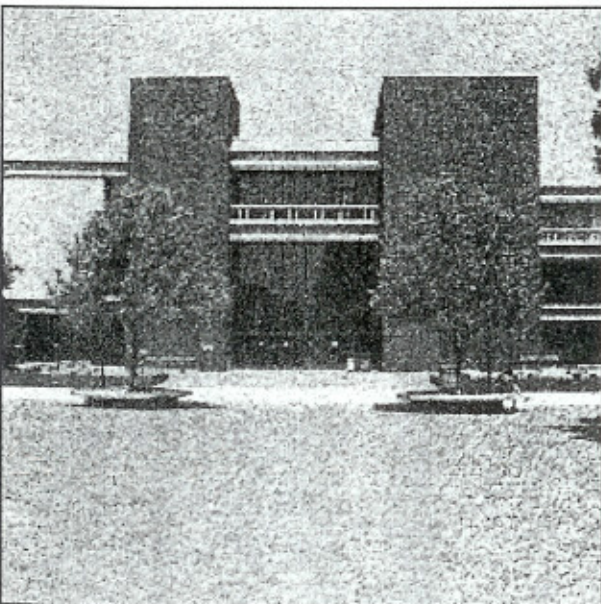
d. Result of geometric mean filter on image with pepper noise; mask size = 3.

mean filter to an image with gaussian noise (a, b) and to an image corrupted with salt and pepper noise (c, d).

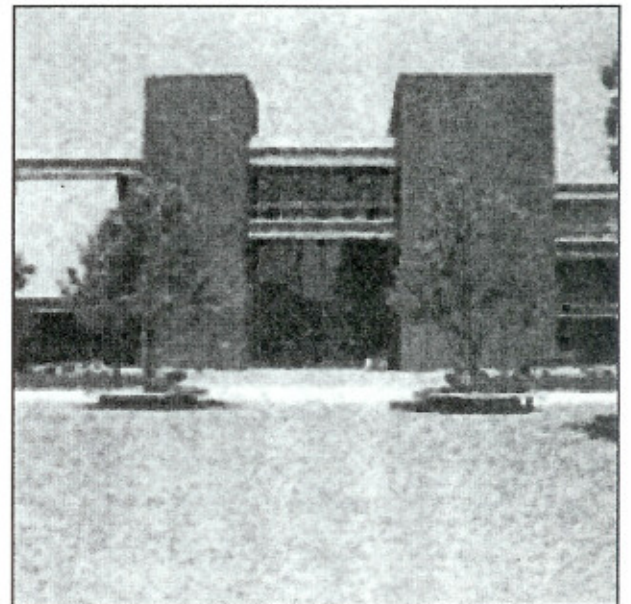
The Y_p mean filter is defined as follows:

$$Y_p \text{ Mean} = \left[\sum_{(r,c) \in W} \frac{d(r,c)^p}{N^2} \right]^{1/p}$$

Figure 3.3-9 Harmonic Mean Filter

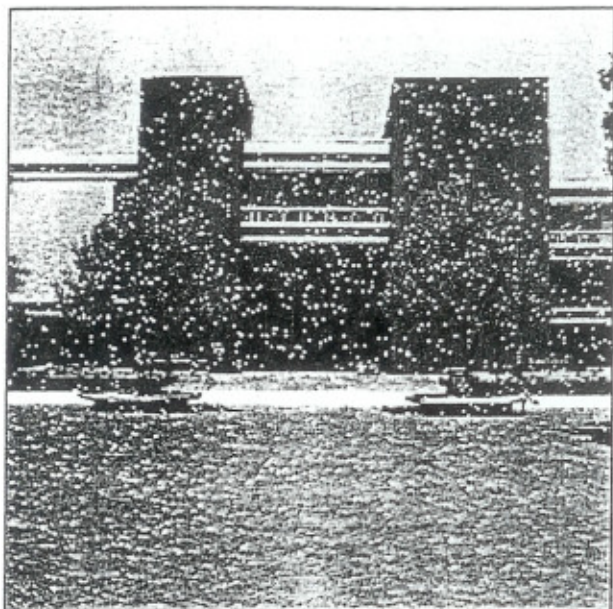


a. Image with gaussian noise—variance = 300; mean = 0.

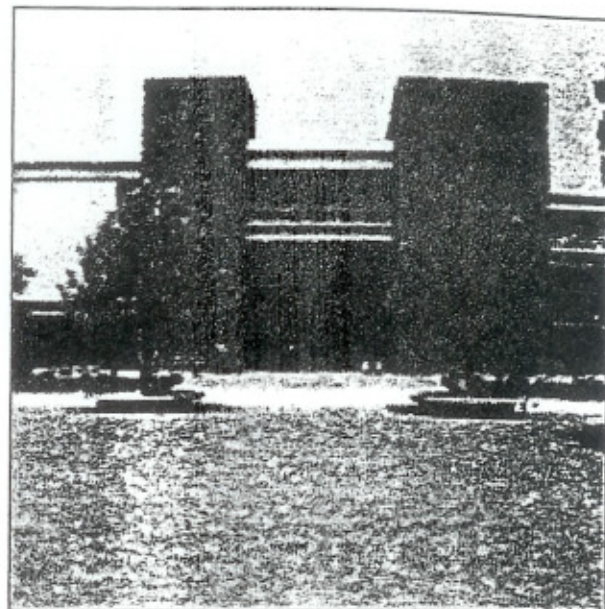


b. Result of harmonic mean filter on image with gaussian noise; mask size = 3.

Figure 3.3-9 (Continued)



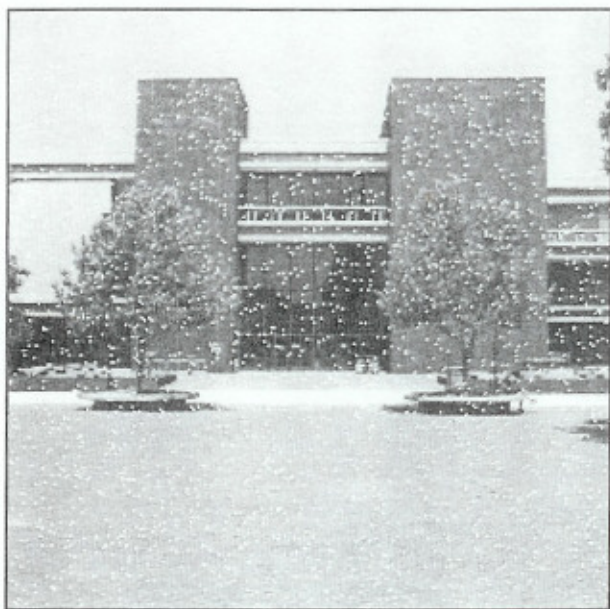
c. Image with salt noise—probability = .04.



d. Result of harmonic mean filter on image with salt noise; mask size = 3.

This filter removes salt noise for negative values of P and pepper noise for negative values of P . Figure 3.3-10 illustrates the use of the Y_p filter.

Figure 3.3-10 Y_p Mean Filter



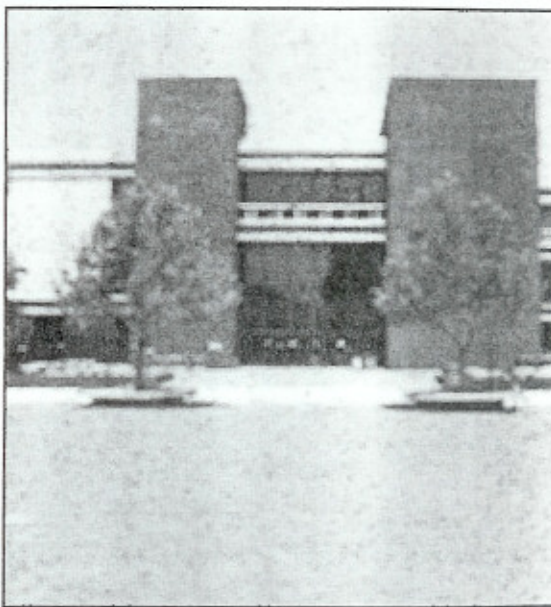
a. Image with salt noise—probability = .04.



b. Result of Y_p mean filter on image with salt noise; mask size = 3, order = -3.



c. Image with pepper noise—probability = .04.



d. Result of Y_p mean filter on image with pepper noise; mask size = 3, order = +3.

Non-linear mean filter

The non-linear mean filter is another generalization of averaging techniques [Pitas and Venetianopoulos 86]; it is defined by

$$f(m, n) = u^{-1} \left\{ \frac{\sum_{(i,j) \in \mathcal{O}} a(i, j) u[g(i, j)]}{\sum_{(i,j) \in \mathcal{O}} a(i, j)} \right\} \quad (4.36)$$

where $f(m, n)$ is the result of the filtering, $g(i, j)$ is the pixel in the input image, and \mathcal{O} is a local neighborhood of the current pixel (m, n) . The function u of one variable has an inverse function u^{-1} ; the $a(i, j)$ are weight coefficients.

If the weights $a(i, j)$ are constant, the filter is called **homomorphic**. Some homomorphic filters used in image processing are

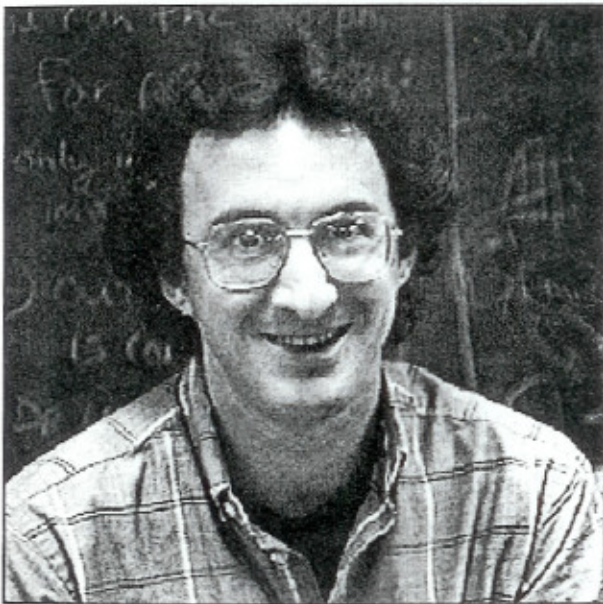
- Arithmetic mean, $u(g) = g$
- Harmonic mean, $u(g) = 1/g$
- Geometric mean, $u(g) = \log g$

Yet another approach to image pre-processing performed in homogeneous pixel neighborhoods is discussed in Section 4.3.9 in the context of several other adaptive-neighborhood pre-processing methods.

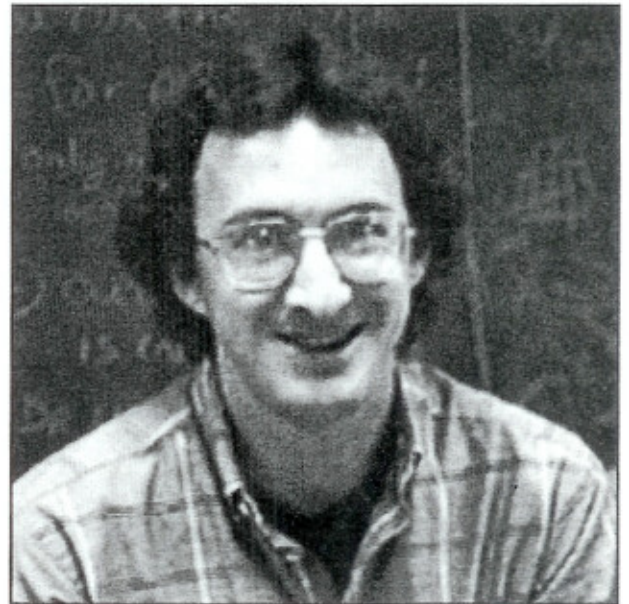
4.4 IMAGE SMOOTHING

Image smoothing is used for two primary purposes: to give an image a softer or special effect or to eliminate noise. In the previous chapter we discussed filtering to eliminate noise, so here we will focus on creating a softer effect (see Image Processing exercise #4 in Chapter 8). Image smoothing is accomplished in the spatial domain by considering a pixel and its neighbors and eliminating any extreme values in this group. This is done by various types of mean and median filters (Chapter 3). In the frequency domain, image smoothing is accomplished by some form of lowpass filtering. Because the high spatial frequencies contain the detail, including edge, information, the elimination of this information via lowpass filtering will provide a smoother image. Any

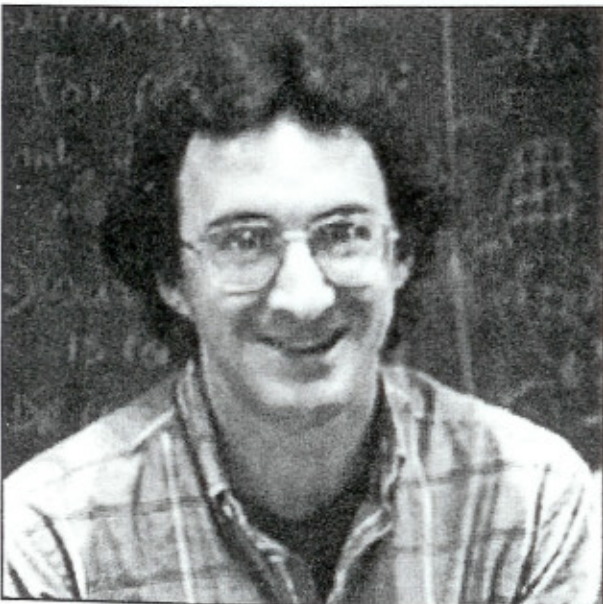
Figure 4.4-1 Mean Filters (3×3)



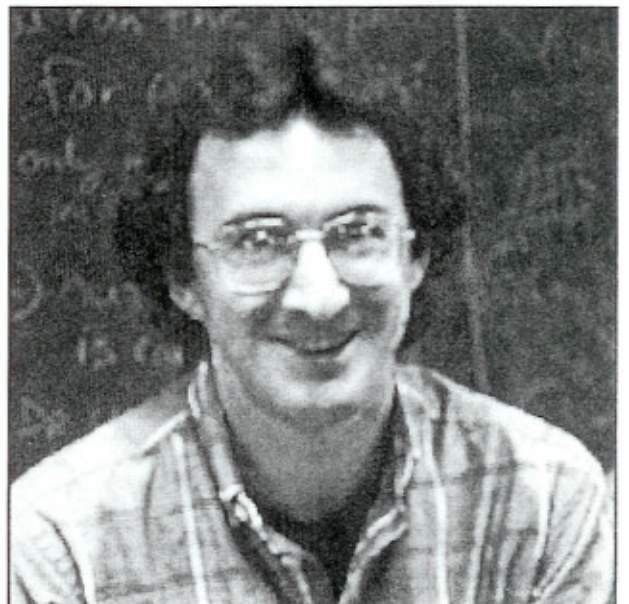
a. Original image.



b. Arithmetic mean filter.



c. Alpha-trimmed mean filter—trim size = 0.



d. Contra-harmonic mean filter—order = +1.

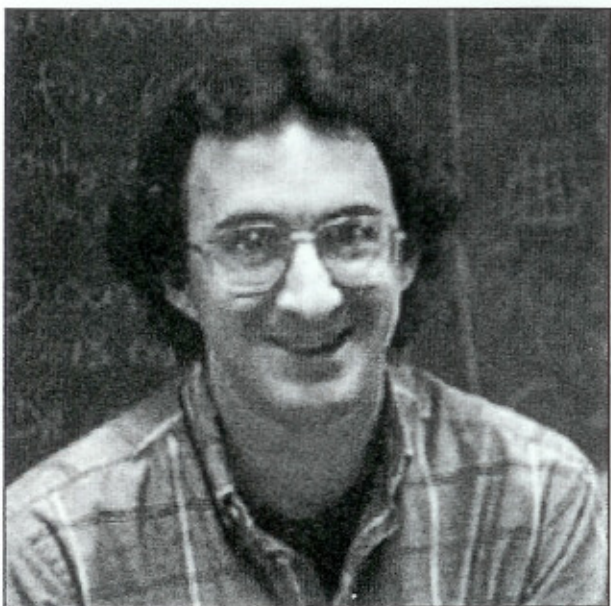
Figure 4.4-1 (Continued)



e. Geometric mean filter.



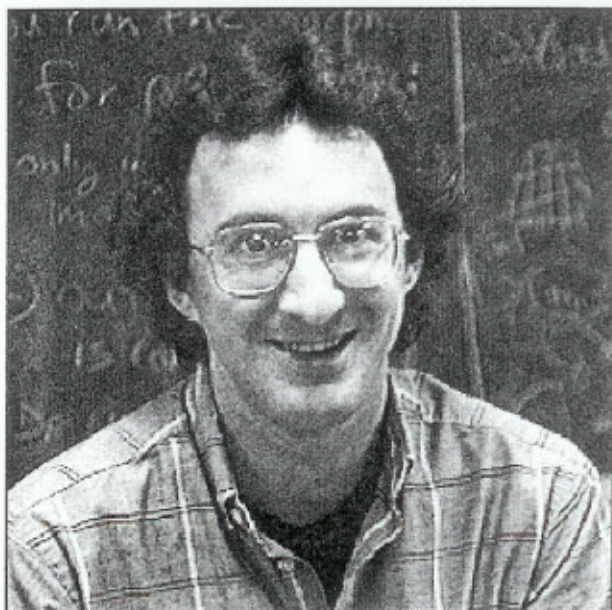
f. Harmonic mean filter.

g. Y_p mean filter—order = +1.

h. Midpoint filter.

fast or sharp transitions in the image brightness will be filtered out, thus providing the desired effect.

Figure 4.4-2 Image Smoothing with an Arithmetic Mean Filter



a. Original image.



b. 3×3 arithmetic mean filter.

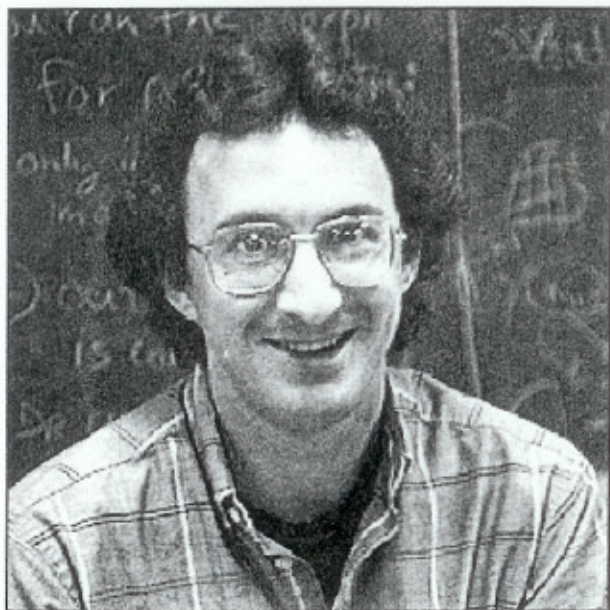


c. 5×5 arithmetic mean filter.

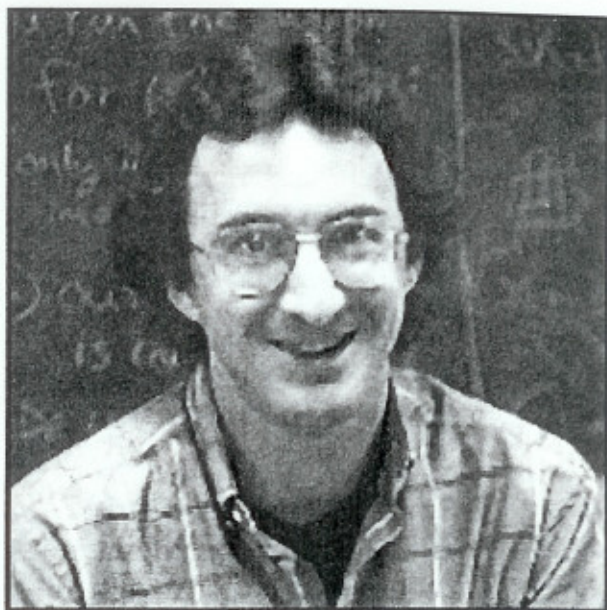


d. 7×7 arithmetic mean filter.

Figure 4.4-3 Image Smoothing with a Median Filter



a. Original image.



b. 3×3 median filter.



c. 5×5 median filter.



d. 7×7 median filter.

data as we move across the image. A pseudo-median filter is explored in the tumor application discussed in Chapter 7.

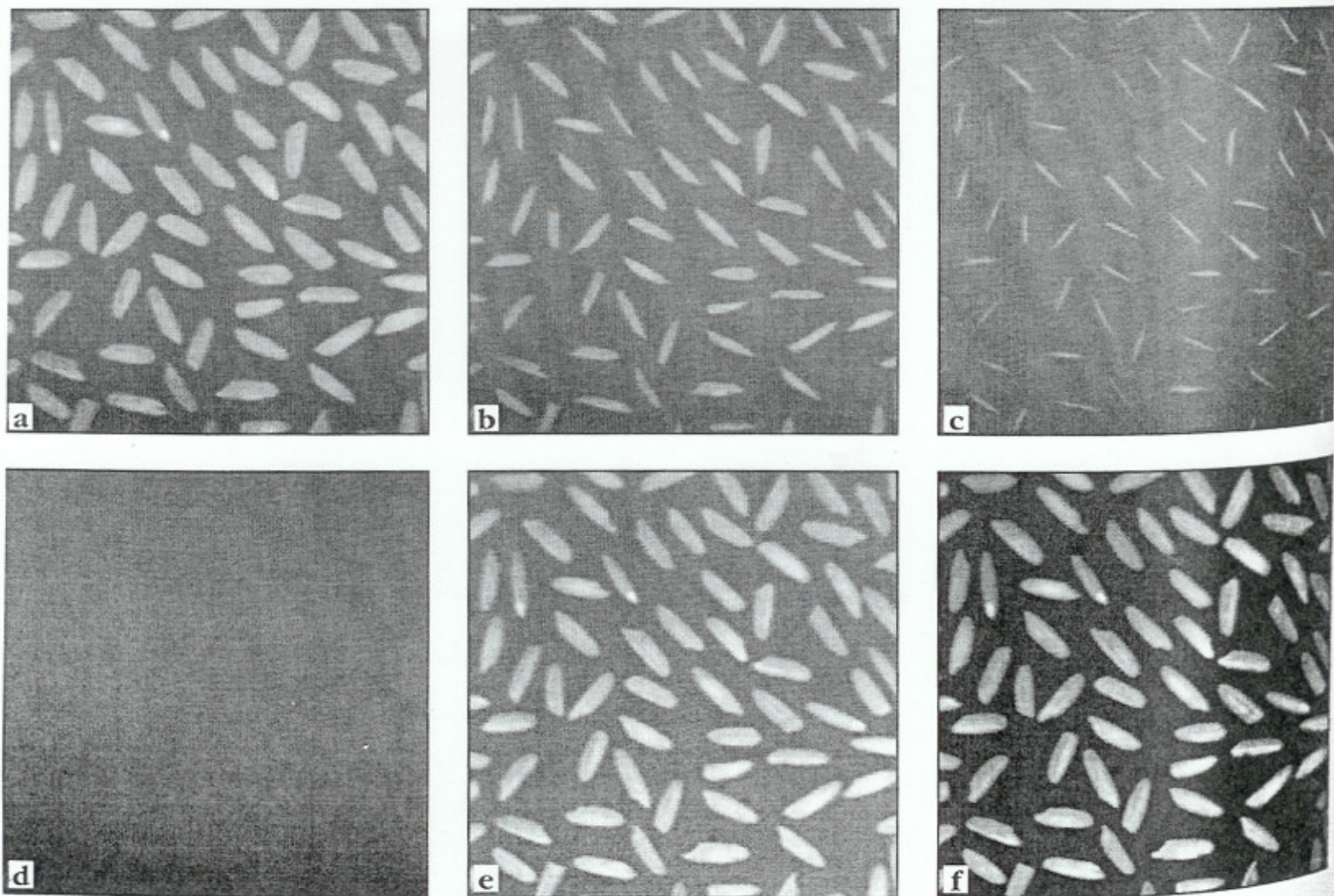


Figure 32. Constructing a background image with a rank operation:
a) an image of rice grains with nonuniform illumination;
b) each pixel replaced with the darkest neighboring pixel in an octagonal 5×5 neighborhood;
c) another repetition of the "darkest neighbor" or grey scale erosion operation;
d) after four repetitions only the background remains;
e) result of subtracting *d* from *a*;
f) the leveled result with contrast expanded.

Weighted Median Filters

Definition: $W \square X = X, X, X, \dots, X$ W times

Definition: median of X_1, X_2, \dots, X_n with weights W_1, W_2, \dots, W_n

$$Y = \text{med}\{W_1 \square X_1, W_2 \square X_2, \dots, W_n \square X_n\}$$

Note: The usual weighted average can be written as

$$Y = \text{mean}\{W_1 \square X_1, W_2 \square X_2, \dots, W_n \square X_n\}$$

Example: $X = [-1 \ 5 \ 8 \ 11 \ -2]$
 $W = [1 \ 2 \ 3 \ 2 \ 1]$

Then $W \square X = \text{med}\{11 \ 11 \ 8 \ 8 \ 8 \ 5 \ 5 \ -1 \ -2\} = 8$

Note that $\text{med}\{X\} = 5$

Theorem: The weighted median chooses β to minimize

$$L(\beta) = \sum_{i=1}^N W_i |x_i - \beta|$$

Note: the arithmetic mean chooses β to minimize $L(\beta) = \sum_{i=1}^N W_i (x_i - \beta)^2$

Calculation of weighted mean

1. Order the x_i
2. add the corresponding w_i from the upper end until half the sum.

$$\text{i.e. } \frac{1}{2} \sum_{i=1}^N w_i$$

3. WM=last sample

Example: $X=[1 \ 5 \ 8 \ 11 \ 2]$ $W=[0.1 \ 0.2 \ 0.3 \ 0.2 \ 0.1]$

1. Sort $X=[11 \ 8 \ 5 \ 2 \ 1]$
 $W=[0.2 \ 0.3 \ 0.2 \ 0.1 \ 0.1]$ $\sum w_i = 0.9$
2. add weights until we get at least half way, i.e. 0.45
3. $0.2 + 0.3 = 0.5 > 0.45$
4. WM=8

As before we can consider a recursive weighted median

Center Weighted Mean Smoother

Choose all the weights = 1 except for the center one which is w_c

$$w_c = 1 \Rightarrow \text{Median}$$

$$w_c \geq N \Rightarrow \text{Identity}$$

Adaptive version: choose $w_c = 1$ for an impulse and $w_c \geq N$ for smooth regions

Negative Weights

Until now we have considered $w_i > 0$

We first begin with averages. We saw that

$$\begin{aligned}\hat{w} &= \text{mean}(w_1 x_1, \dots, w_N x_N) \\ &= \text{mean}(|w_1| \text{sign}(w_1) x_1, \dots, |w_N| \text{sign}(w_N) x_N)\end{aligned}$$

So analogously we define the weighted median as

$$\begin{aligned}\text{weighted-med}(w) &= \text{median}(w_1 x_1, \dots, w_N x_N) \\ &= \text{median}(|w_1| (\text{sign}(w_1) x_1), \dots, |w_N| (\text{sign}(w_N) x_N))\end{aligned}$$

Computation:

1. $T_0 = \frac{1}{2} \sum_{i=1}^N |w_i|$
2. sort $y_i = \text{sgn}(w_i) x_i$
3. sum w_i corresponding to y_i beginning with largest values
4. output is signed sample which causes sum to be greater or equal to T_0

negative weighted median

$$\begin{array}{ccc} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{array}$$

Let g_c be the gray value of the central pixel. We order the gray values as

$$- g_{\text{largest}}, \dots, - g_{\text{smallest}}, \underbrace{g_c, g_c, \dots, g_c}_8$$

Since we have an even number (16) of values we take the average of the two middle ones

$$WM = \frac{g_c - g_{\text{smallest}}}{2}$$

Note: we get different results if we go from black to white or white to black

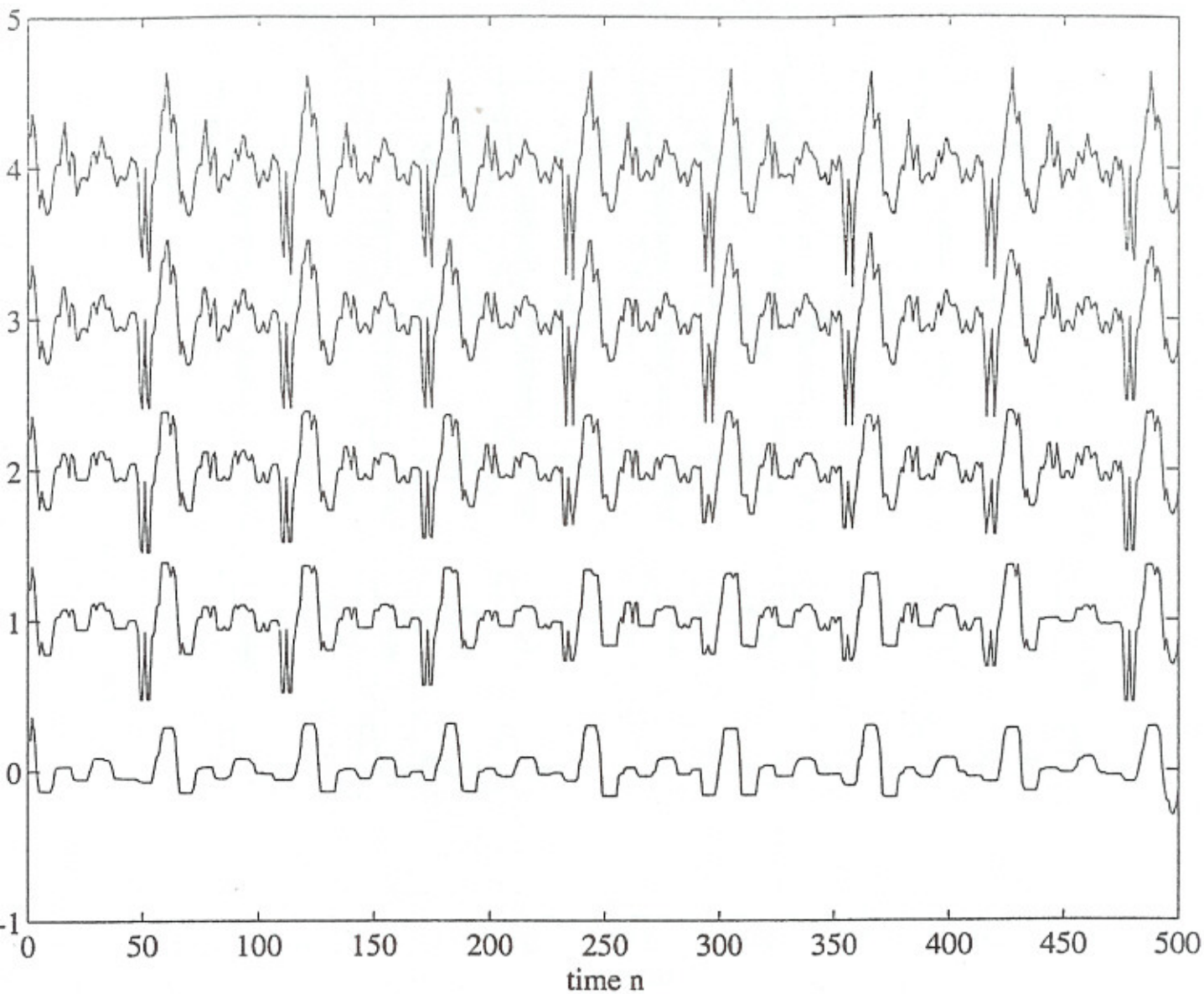


FIGURE 3 Effects of increasing the center weight of a CWM smoother of window size $N = 9$ operating on the voiced speech "a". The CWM smoother output is shown for $W_c = 2w + 1$, with $w = 0, 1, 2, 3$. Note that for $W_c = 1$ the CWM reduces to median smoothing, and for $W_c = 9$ it becomes the identity operator.

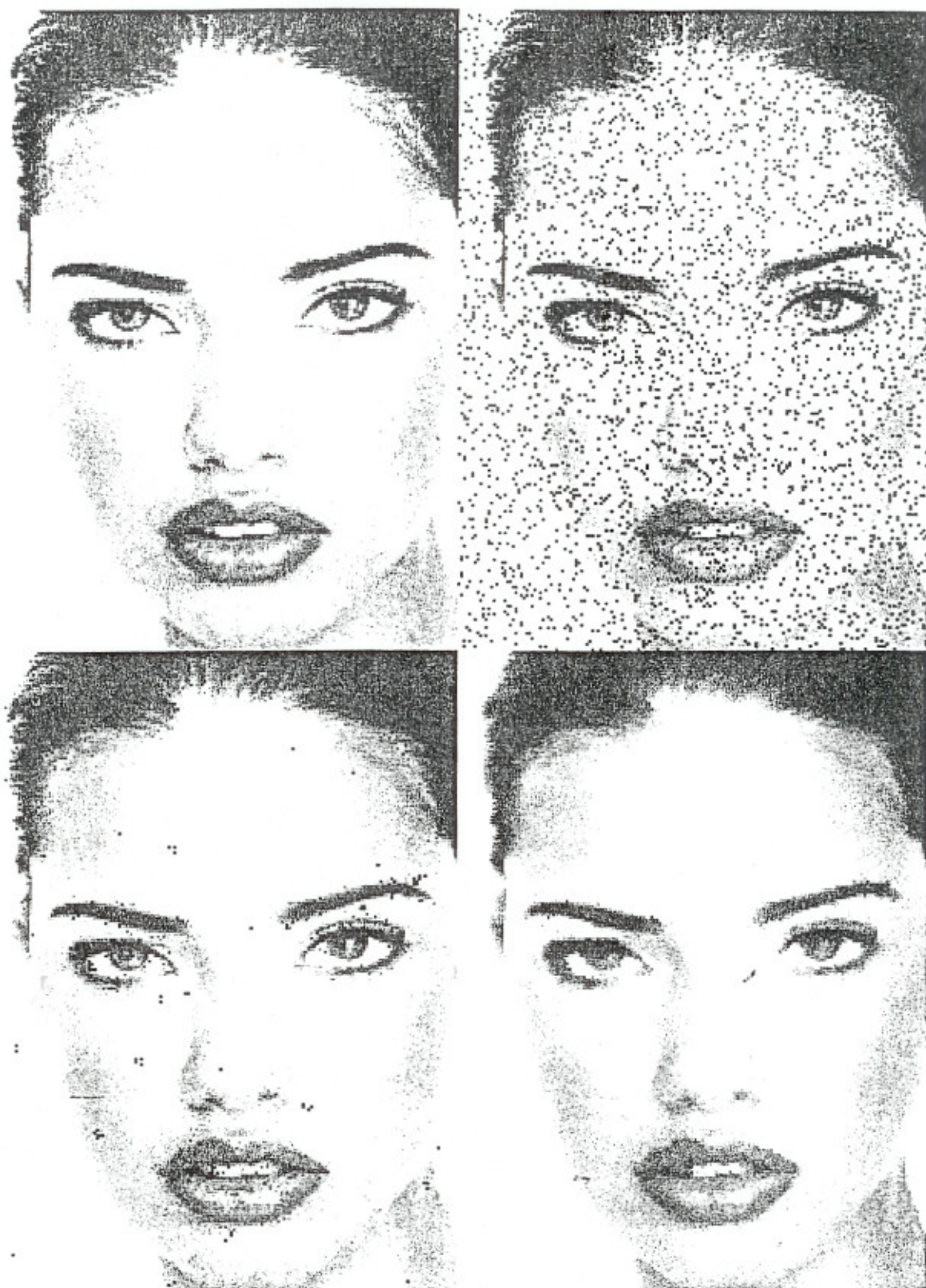


FIGURE 9 Impulse noise cleaning with a 5×5 CWM smoother: (a) original gray-scale "portrait" image, (b) image with salt-and-pepper noise, (c) CWM smoother with $W_c = 15$, (d) CWM smoother with $W_c = 5$.

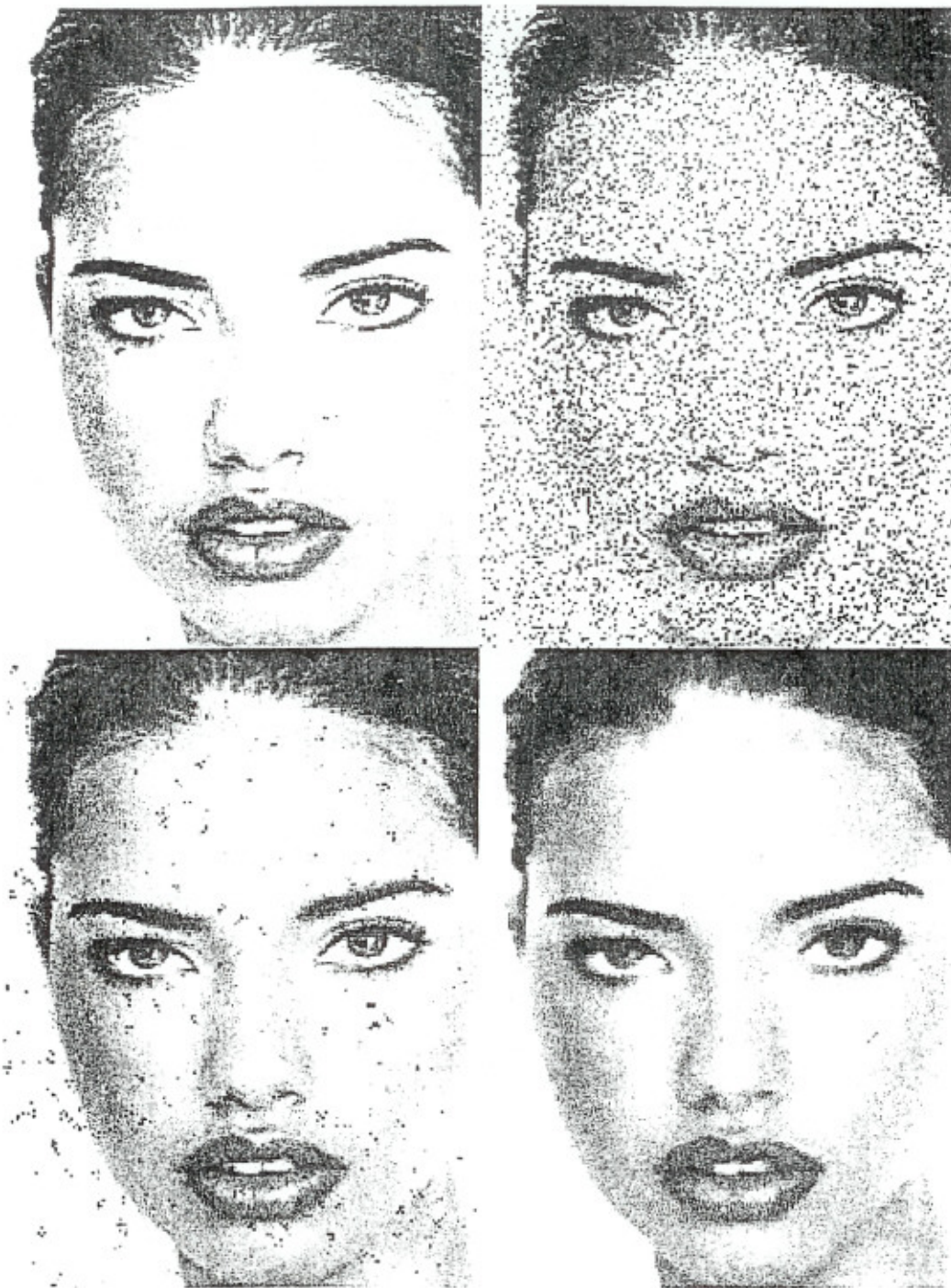


FIGURE 10 Impulse noise cleaning with a 5×5 CWM smoother: (a) original "portrait" image, (b) image with salt-and-pepper noise, (c) CWM smoother with $W_c = 16$, (d) CWM smoother with $W_c = 5$. (See color section, p. C-3.)



FIGURE 11 (Enlarged) Noise-free image (left), 5×5 median smoother output (center), and 5×5 mean smoother (right). (See color section, p. C-4.)



FIGURE 12 (Enlarged) CWM smoother output (left), recursive CWM smoother output (center), and permutation CWM smoother output (right). Window size is 5×5 . (See color section, p. C-4.)

NAME medtrunc

DESCRIPTION

Performs the Truncated median filter. This filter is an approximation of the 'mode filter'. The mode of the distribution of brightness values in each neighborhood is, by definition, the most likely value. However, for a small neighborhood, the mode is poorly defined. For any asymmetric distribution, such as

would be obtained at most locations near but not precisely straddling an edge, the mode is the highest point, and the median lies closer to the mode than the mean value. The truncated median filter first calculates the mean in each 3x3 region. Then it calculates the median of the 7 values in each 3x3 region which are closest to the mean value.

This has the effect of sharpening steps, and produces posterization when it is applied repeatedly.

EXAMPLE



Original noisy Image



Truncated median filter

Execution time: 3.224ms (RVS-10G), 2.399ms (RVS-DX-10G)

NAME medhybrid

DESCRIPTION

Performs the Hybrid Median filter. This is an edge-preserving median filter. It's a three-step ranking operation. In a 5x5 area pixels are ranked in two different groups as shown below. The median value of group 'a' and 'b' and the central pixel 'ab' is used as the new pixel value.

This filter can be used to remove "shot" noise (in which individual pixels are corrupted or missing from an image) or to reduce random noise in the context of averaging. This median filter overcomes the tendency of the other median filters (median3, mediannn, medtrunc) to erase lines which are narrower than the half-width of the neighborhood and to round corners.

```
aa  bb  aa
  aa bb aa
bb bb ab bb bb
  aa bb aa
aa  bb  aa
```

EXAMPLE



Original Image



Hybrid Median filter

Non-Linear Filters

A variety of smoothing filters have been developed that are not linear. While they cannot, in general, be submitted to Fourier analysis, their properties and domains of application have been studied extensively.

* *Median filter* - The median statistic was described in Section 3.5.2. A median filter is based upon moving a window over an image (as in a convolution) and computing the output pixel as the median value of the brightnesses within the input window. If the window is $J \times K$ in size we can order the $J \times K$ pixels in brightness value from smallest to largest. If $J \times K$ is odd then the median will be the $(J \times K + 1)/2$ entry in the list of ordered brightnesses. Note that the value selected will be exactly equal to one of the existing brightnesses so that no roundoff error will be involved if we want to work exclusively with integer brightness values. The algorithm as described above has a generic complexity per pixel of $O(J \times K \log(J \times K))$. Fortunately, a fast algorithm (due to J. A. Stiles et al.) exists that reduces the complexity to $O(K)$ assuming $J \geq K$.

A useful variation on the theme of the median filter is the *percentile filter*. Here the center pixel in the window is replaced not by the 50% (median) brightness value but rather by the $p\%$ brightness value where $p\%$ ranges from 0% (the *minimum filter*) to 100% (the *maximum filter*). Values other than ($p=50$)% do not, in general, correspond to smoothing filters.

* *Kuwahara filter* - Edges play an important role in our perception of images (see Figure 15) as well as in the analysis of images. As such it is important to be able to smooth images without disturbing the sharpness and, if possible, the position of edges. A filter that accomplishes this goal is termed an *edge-preserving filter* and one particular example is the Kuwahara filter. Although this filter can be implemented for a variety of different window shapes, the algorithm will be described for a square window of size $J = K = 4L + 1$ where L is an integer. The window is partitioned into four regions as shown in Figure 29.

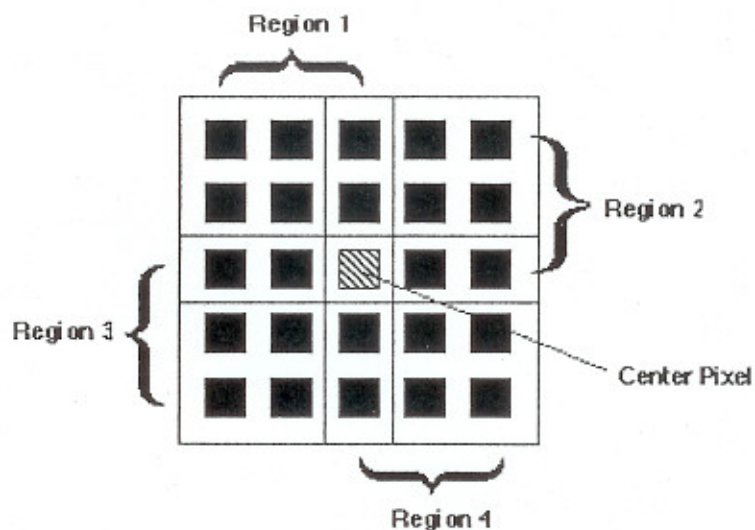


Figure 29: Four, square regions defined for the Kuwahara filter. In this example $L=1$ and thus $J=K=5$. Each region is $[(J+1)/2] \times [(K+1)/2]$.

In each of the four regions ($i=1,2,3,4$), the mean brightness, m_i in eq. , and the variance, s_i^2 in eq. , are measured. The output value of the center pixel in the window is the mean value of that region that has the smallest variance.

NAME kuwahara

DESCRIPTION

Performs the Kuwahara Filter. This filter is an edge-preserving filter.

(a a ab b b)
(a a ab b b)
(ac ac abcd bd bd)
(c c cd d d)
(c c cd d d)

In each of the four regions (a, b, c, d), the mean brightness and the variance are calculated. The output value of the center pixel (abcd) in the window is the mean value of that region that has the smallest variance.

This filter is an edge-preserving filter, which smooths the images without disturbing the sharpness and the position of edges.

EXAMPLE



Original Image



Img after Kuwahara Filter



3.3.3 Adaptive Filters—Minimum Mean-Square Error Filter

The previously described filters are adaptive in the sense that their output depends on the underlying pixel values. Some, such as the alpha-trimmed mean, can vary between a mean and median filter, but this change in filter behavior is fixed for a given value of the T parameter. However, an *adaptive filter* alters its basic behavior as the image is processed; it may act like a mean filter on some parts of the image and a median filter on other parts of the image. The typical criteria used to determine the filter behavior are the local image characteristics, usually measured by the local gray level statistics. The minimum mean-square error (MMSE) filter is a good example of an adaptive filter, which exhibits varying behavior based on local image statistics. The MMSE filter works best with gaussian or uniform noise and is defined as follows:

$$\text{MMSE} = d(r, c) - \frac{\sigma_n^2}{\sigma_l^2} [d(r, c) - m_l(r, c)]$$

where

σ_n^2 = noise variance

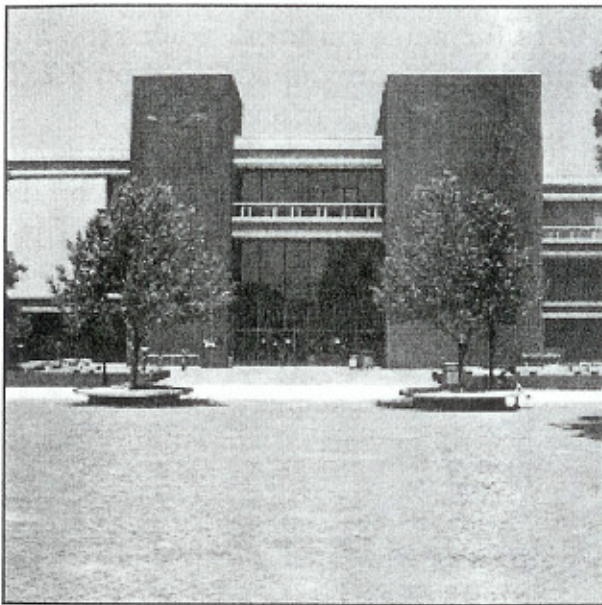
σ_l^2 = local variance (in the window under consideration)

m_l = local mean (average in the window under consideration)

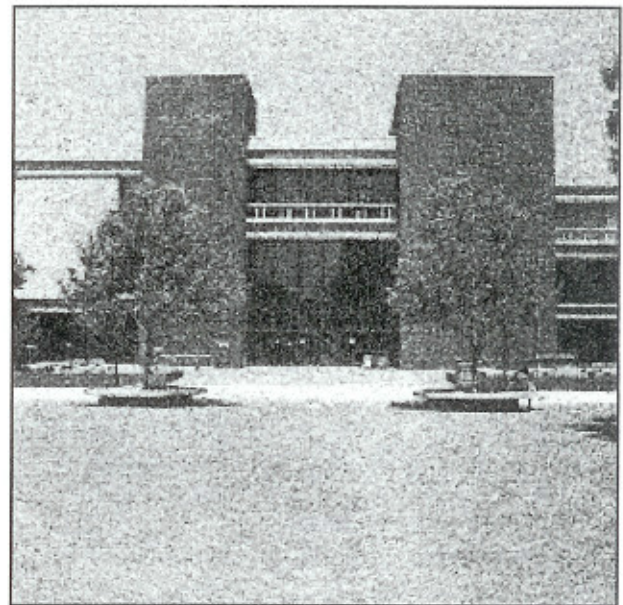
With no noise in the image, the noise variance equals zero, and this equation will return the original unfiltered image. In background regions of the image, areas of

are weighted by the noise to local variance ratio, σ_n^2/σ_i^2 . As this ratio increases, implying primarily noise in the window, the filter returns primarily the local average. As this ratio goes down, implying high local detail, the filter returns more of the original unfiltered image. By operating in this manner, the MMSE filter adapts itself to the local image statistics, preserving image details while removing noise. Figure 3.3-11 illustrates the use of the MMSE filter on an image with added gaussian noise. Here we specify the window (kernel) size and the noise variance to be used. More information on adaptive filters can be found in the references.

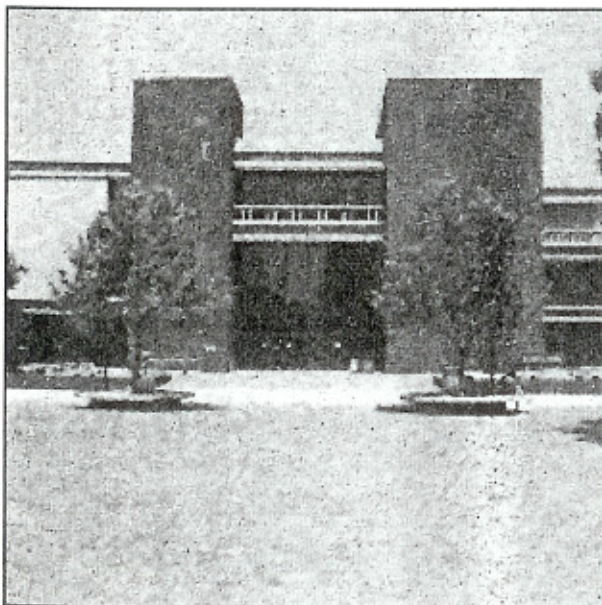
Figure 3.3-11 MMSE Filter



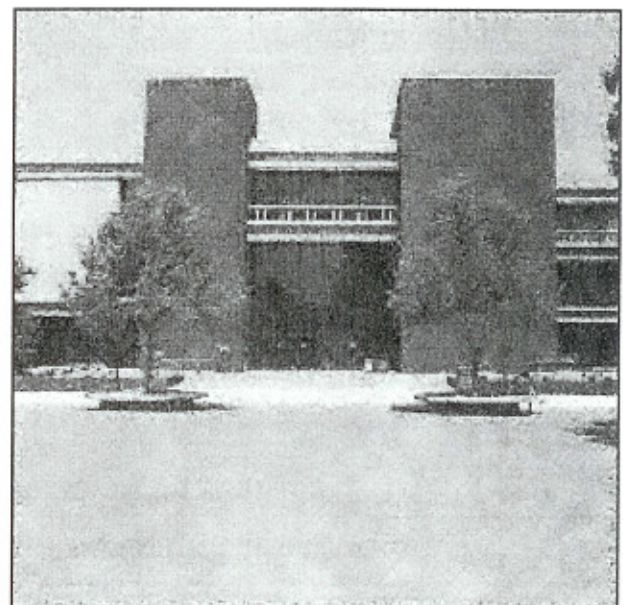
a. Original image.



b. Image with gaussian noise—variance = 300; mean = 0.



c. Result of MMSE filter—kernel size = 3; noise variance = 300.



d. Result of MMSE filter—kernel size = 9; noise variance = 300.

Averaging using a rotating mask

Averaging using a rotating mask is a method that avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood and the resulting image is in fact sharpened [Nagao and Matsuyama 80]. The brightness average is calculated only within this region; a brightness dispersion σ^2 is used as the region homogeneity measure. Let n be the number of pixels in a region R and g be the input image. Dispersion σ^2 is calculated as

$$\sigma^2 = \frac{1}{n} \left\{ \sum_{(i,j) \in R} \left[g(i,j) - \frac{1}{n} \sum_{(i,j) \in R} g(i,j) \right]^2 \right\} \quad (4.33)$$

The computational complexity (number of multiplications) of the dispersion calculation can be reduced if equation (4.33) is expressed another way:

$$\begin{aligned} \sigma^2 &= \frac{1}{n} \sum_{(i,j) \in R} \left\{ [g(i,j)]^2 - 2g(i,j) \frac{\sum_{(i,j) \in R} g(i,j)}{n} + \left[\frac{\sum_{(i,j) \in R} g(i,j)}{n} \right]^2 \right\} \\ &= \frac{1}{n} \left\{ \sum_{(i,j) \in R} [g(i,j)]^2 - 2 \frac{[\sum_{(i,j) \in R} g(i,j)]^2}{n} + n \left[\frac{\sum_{(i,j) \in R} g(i,j)}{n} \right]^2 \right\} \\ &= \frac{1}{n} \left\{ \sum_{(i,j) \in R} [g(i,j)]^2 - \frac{[\sum_{(i,j) \in R} g(i,j)]^2}{n} \right\} \end{aligned} \quad (4.34)$$

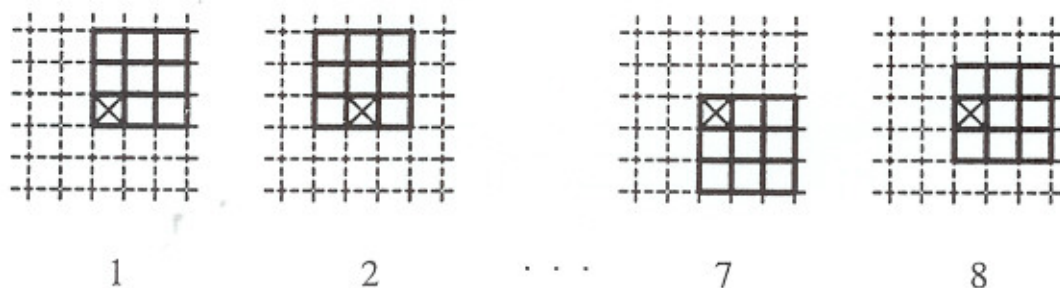


Figure 4.12: Eight possible rotated 3×3 masks.

Having computed region homogeneity, we consider its shape and size. The eight possible 3×3 masks that cover a 5×5 neighborhood of a current pixel (marked by the small cross) are

shown in Figure 4.12. The ninth mask is the 3×3 neighborhood of the current pixel itself. Other mask shapes can also be used. Figure 4.13 shows another set of eight masks covering a 5×5 neighborhood of the current pixel. Again the ninth mask is the 3×3 neighborhood of the current pixel. Another possibility is to rotate a small 2×1 mask to cover the 3×3

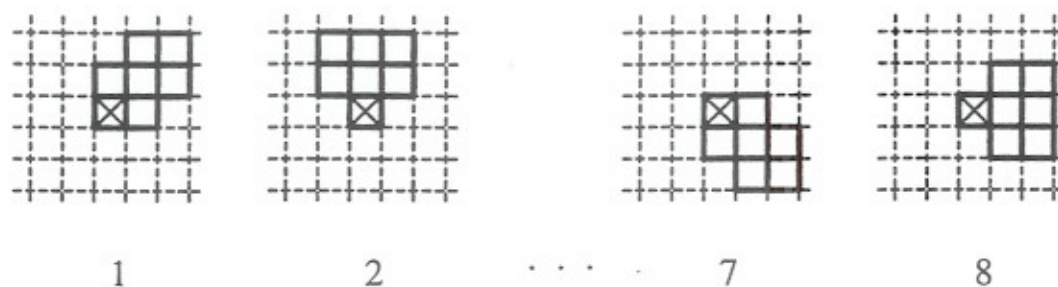


Figure 4.13: *Alternative shape of eight possible rotated masks.*

neighborhood of the current pixel.

Image smoothing using the rotating mask technique uses the following algorithm.

Algorithm 4.2: Smoothing using a rotating mask

1. Consider each image pixel (i, j) .
2. Calculate dispersion in the mask for all possible mask rotations about pixel (i, j) according to equation (4.33).
3. Choose the mask with minimum dispersion.
4. Assign to the pixel $f(i, j)$ in the output image f the average brightness in the chosen mask.

Algorithm 4.2 can be used iteratively; the iterative process converges quite quickly to the stable state (that is, the image does not change any more). The size and shape of masks influence the convergence—the smaller the mask, the smaller are the changes and more iterations are needed. A larger mask suppresses noise faster and the sharpening effect is stronger. On the other hand, information about details smaller than the mask may be lost. The number of iterations is also influenced by the shape of regions in the image and noise properties.

Bilateral Filter

in one dimension use weighted average

$$\hat{x}_k = \frac{\sum_n w_{kn} y_{k-n}}{\sum_n w_{kn}}$$

where

1. $w_{kn}^S = e^{-\frac{d_{k,k-n}^2}{2\sigma_s^2}}$ physical distance

2. $w_{kn}^R = e^{-\frac{d^2(y_k, y_{k-n})}{2\sigma_s^2}} = e^{-\frac{(y_k - y_{k-n})^2}{2\sigma_s^2}}$ gray level distance

Then $w_{k,n} = w_{k,n}^S \cdot w_{k,n}^R$

Note: when

σ_s and σ_R are large \Rightarrow uniform non-adaptive; degrades signal

σ_s and σ_R are small \Rightarrow no smoothing

Nonlinear Filters: Bilateral Filter

- Combines the idea of weighted filtering with an outlier rejection
- Similar to Gaussian filter, it is also defined as a weighted average of pixels.
- The difference is that the bilateral filter takes into account the variation of intensities to preserve edges.

Nonlinear Filters: Bilateral Filter

- The output pixel value again depends on a weighted combination

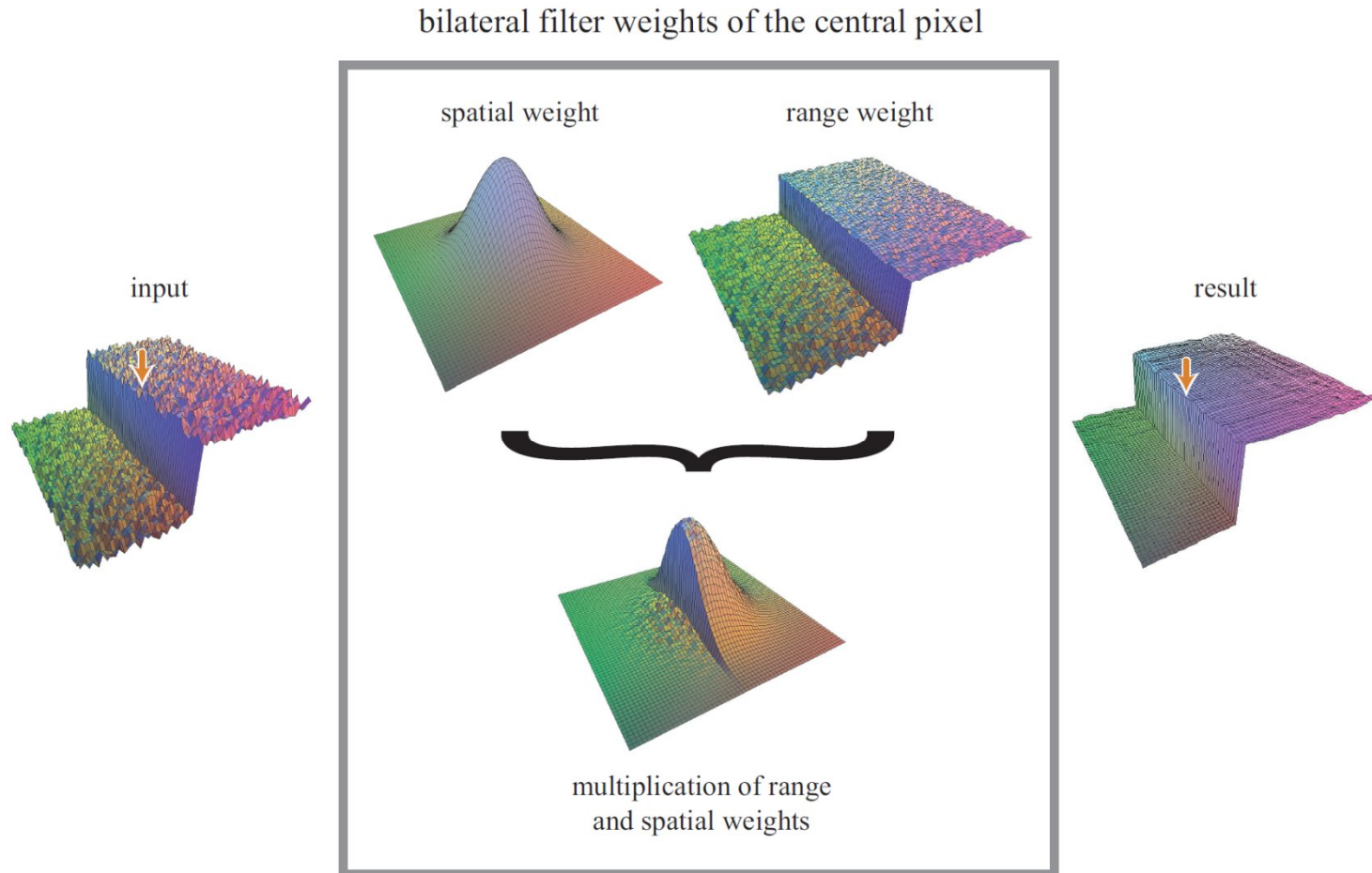
$$g(i, j) = \frac{\sum_{k,l} f(k, l)w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)}$$

$$w(i, j, k, l) = \exp \left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2} - \frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2} \right)$$

w is a product of two weights: domain and range.

- Domain component is a usual Gaussian which penalizes distant pixels
- Range component penalizes pixels with a different intensity

Nonlinear Filters: Bilateral Filter



Results

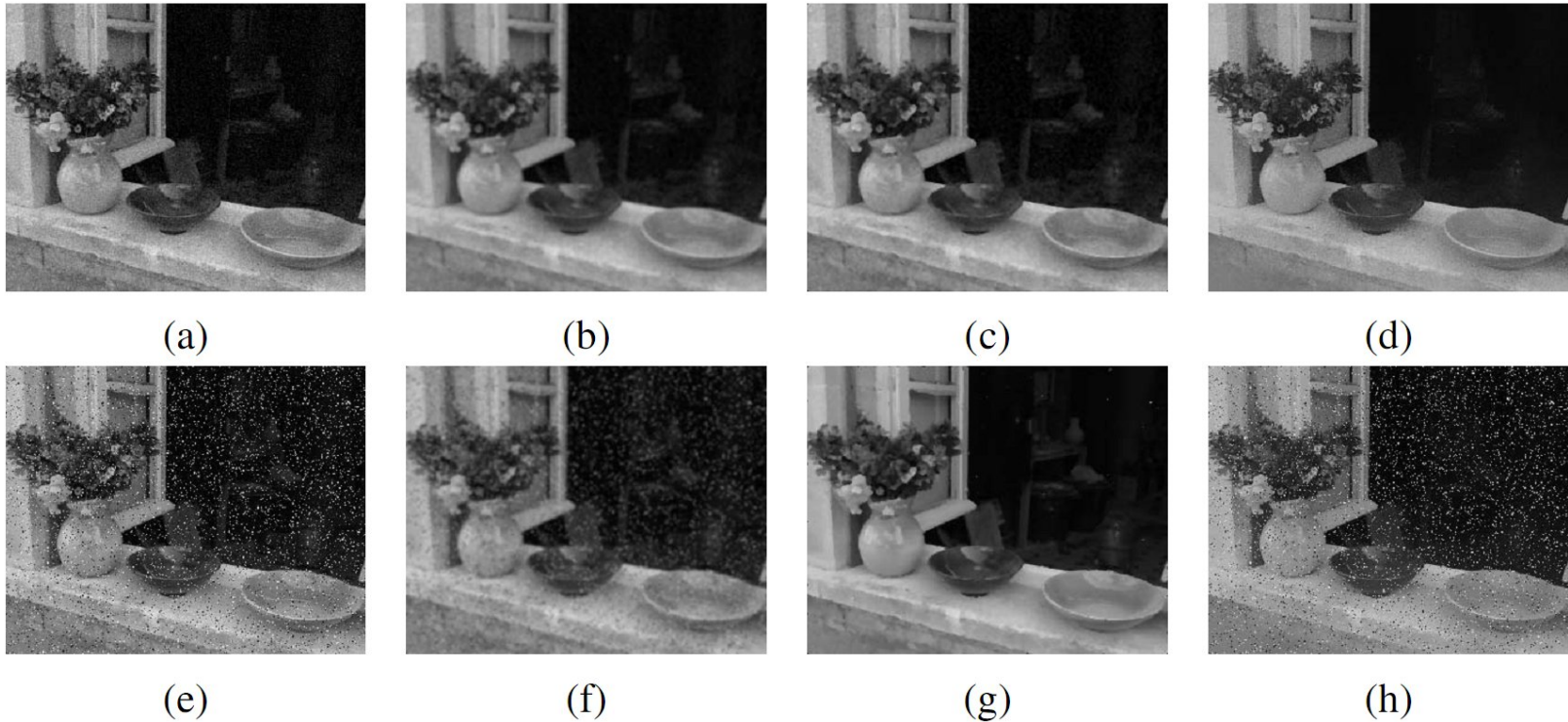
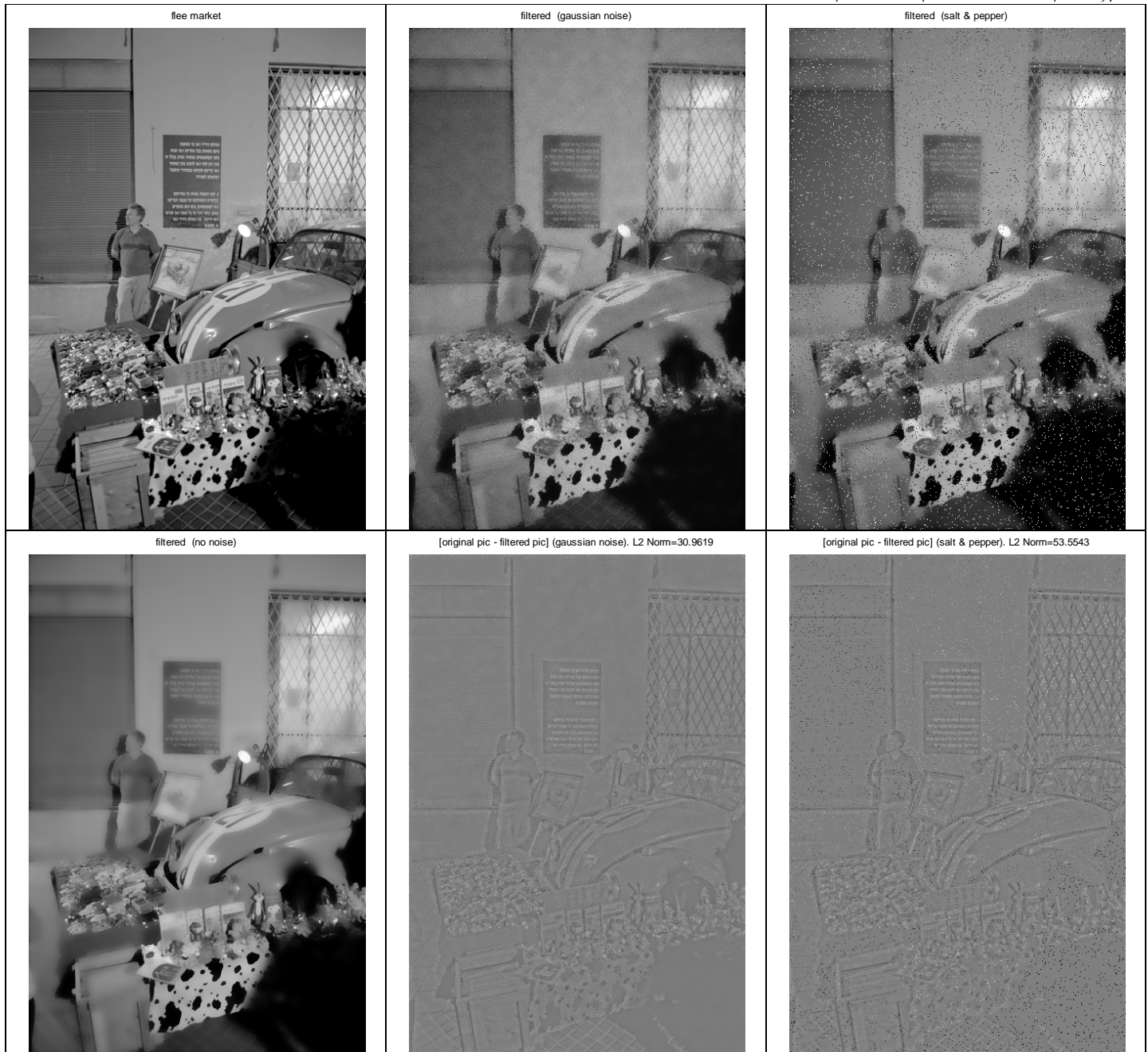


Figure 3.17: *Median and bilateral filtering: (a) image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that for shot noise, the bilateral filter fails to remove the noise, because the noisy pixels are too different from their neighbors to get filtered.*

פילטר bilateral

נבחן הפעלת פילטר bilateral בעל $\sigma=7$ במרחב, ו- $\sigma=0.2$ בצבע.

ניתן לראות, כי הפילטר על התמונה המקורית מצליח לבצע טשטוש תוך כדי שמירה על קצוות חדים, זאת למרות ה- σ הגדולה במרחב. עדיין, קצוות שאינם חדים מיטשטשים (תווי הפנים של האיש, הציור והטקסטורות העדינות של הצעצועים). הדבר עוזר להפחית רעש גאוסיאני, אך אינו עוזר להתמודד עם מלח-פלפל. זאת כיוון שערכי הפיקסלים בהם יש מלח או פלפל רחוקים מערכי הפיקסלים סביבם. לכן, הפיקסלים סביבם מקבלים משקולות נמוכות ואינם משפיעים עליהם.



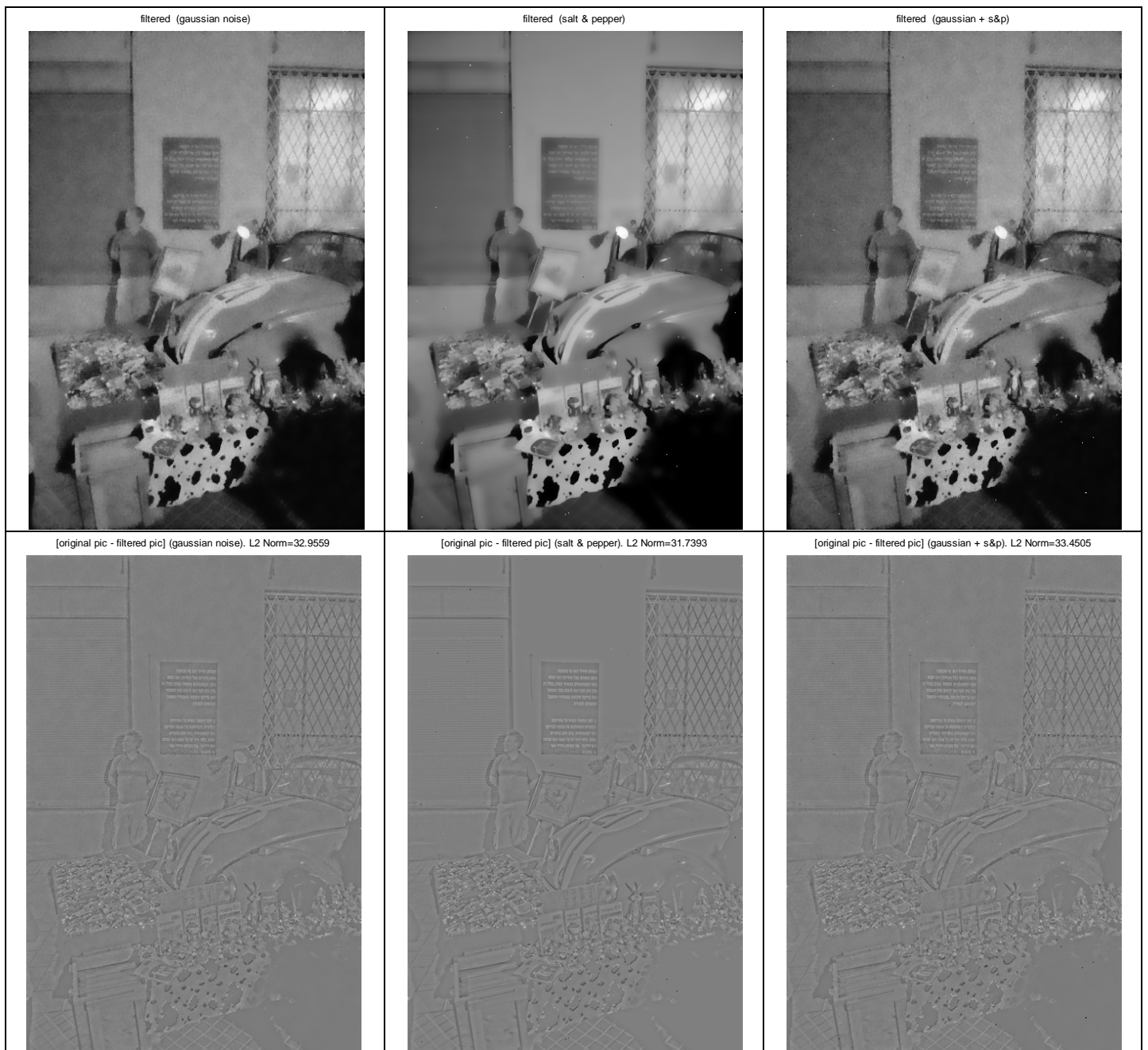
פילטר bilateral משופר

על מנת לשפר את ההתמודדות של פילטר bilateral עם רעש מלח-פלפל, נקצה את המשקולות עבור ערך רמת האפור, לא ע"פ מרחקם מהפיקסל המרכזי, אלא מרחקם מהחציון. במקרה זה נבחר חציון של סביבה קטנה יחסית – ארבעת הפיקסלים הצמודים לפיקסל המרכזי. הנוסחה עבור משקולות הערך $w_{k,n}^R$ (בהתאם לסימונים שהוצגו בכיתה) היא במקרה זה:

$$w_{k,n}^R = \exp\left(-\frac{\text{median}(\text{neighbours of } Y_k) - Y_{k-n}}{2\sigma^2}\right)$$

כפי שראינו, חציון של סביבה אינו מוטה מקיומם של רעשים קיצוניים ויחד עם זאת שומר על קצוות חדים בתמונה. לכן, שימוש במשקולות אלה יאפשר לתת משקל לסביבה של פיקסלי מלח/פלפל.

כפי שניתן להתרשם, שיפור זה מאפשר להפחית רעשים גאוסיינים ורעשי מלח פלפל, ונותן תוצאות טובות גם בתמונה מורעשת ברעש משולב.



Weighted Least Squares (WLS)

Let Y = measured image

X = desired image

then

$$\varepsilon_{WLS}(X) = \frac{1}{2}(X - Y)^T(X - Y) + \underbrace{\frac{\lambda}{2}(X - DX)^T(X - DX)}_{\text{penalty}}$$

If D is left shift the $X - DX$ is a first difference

Choose W diagonal

Robust Estimation

$$\varepsilon_{RE} = \frac{1}{2}(X - Y)^T(X - Y) + \frac{\lambda}{2}\rho(X - DX)$$

Averaging according to inverse gradient

The convolution mask is calculated at each pixel according to the inverse gradient [Wang and Vagnucci 81], the idea being that the brightness change within a region is usually smaller than between neighboring regions. Let pixel location (m, n) correspond to that of a central pixel of a convolution mask with odd size; the inverse gradient δ at the point (i, j) with respect to (m, n) is then

$$\delta(i, j) = \frac{1}{|g(m, n) - g(i, j)|} \quad (4.31)$$

If $g(m, n) = g(i, j)$, then we define $\delta(i, j) = 2$; the inverse gradient δ is then in the interval $(0, 2]$, and δ is smaller on the edge than in the interior of a homogeneous region. Weight coefficients in the convolution mask h are normalized by the inverse gradient, and the whole term is multiplied by 0.5 to keep brightness values in the original range. The constant 0.5 has the effect of assigning half the weight to the central pixel (m, n) , and the other half to neighborhood.

$$h(i, j) = 0.5 \frac{\delta(i, j)}{\sum_{(m, n) \in \mathcal{O}} \delta(i, j)} \quad (4.32)$$

The convolution mask coefficient corresponding to the central pixel is defined as $h(i, j) = 0.5$.

This method assumes sharp edges. When the convolution mask is close to an edge, pixels in the region have larger coefficients than pixels near the edge, and it is not blurred. Isolated noise points within homogeneous regions have small values of the inverse gradient; pixels from the neighborhood take part in averaging and the noise is removed.

L1 Filters

$$\bar{x} = (x_1, x_2, \dots, x_n)$$

reform as an ordered sequence

$$\bar{z} = (x_{(1)}, x_{(2)}, \dots, x_{(n)})$$

where

$$x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$$

Then

$$d = \sum_{i=1}^M c_i z_i$$

If $z_i = x_i$ and $M = n$ then we have a linear estimator

This filter includes the median, alpha-trim, midpoint, etc.

Let the weight c_i depend on the distance from the median. Then define the N^2 dimensional vector

$$z = \begin{matrix} x_{(1),1} & \dots & x_{(1),N} \\ x_{(2),1} & \dots & x_{(2),N} \\ \dots & & \\ x_{(N),1} & \dots & x_{(N),N} \end{matrix}$$

where

$$x_{(i),j} = \begin{cases} x_j & \text{j-th element in i-th rank} \\ 0 & \text{otherwise} \end{cases}$$

example:

$$\bar{x} = (1, 3, 1, 2)$$

then

$$\bar{x}_r = (1, 1, 2, 3)$$

and

$$z = \begin{matrix} 1000 \\ 0010 \\ 0002 \\ 0300 \end{matrix}$$

Now do linear estimation on z

Recursive Filters

$$g'_n = \underbrace{-\sum_{l=1}^s a_l g'_{n-l}}_{IIR} + \underbrace{\sum_{l=-R}^R h_l g_{n-l}}_{FIR}$$

or

$$\sum_{l=0}^s a_l g'_{n-l} = \sum_{l=-R}^R h_l g_{n-l} \quad a_0 = 1$$

Taking the Fourier transform we get

$$\sum_{l=0}^s a_l e^{-2\pi i k l} \widehat{g}'(k) = \sum_{l=-R}^R h_l e^{-2\pi i k l} \widehat{g}(k)$$

Transfer Function

$$\widehat{h}(k) = \frac{\widehat{g}'(k)}{\widehat{g}(k)} = \frac{\sum_{l=-R}^R h_l e^{-2\pi i k l}}{\sum_{l=0}^s a_l e^{-2\pi i k l}}$$

Define $z = re^{2\pi i k}$

Then

$$\widehat{h}(k) = \frac{\sum_{l=-R}^R h_l z^{-l}}{\sum_{l=0}^s a_l z^{-l}} = z^{s-R} \frac{\sum_{l=-R}^R h_l z^{R-l}}{\sum_{l=0}^s a_l z^{s-l}}$$

A polynomial of degree p has p complex roots.

Hence,

$$\sum_{l=0}^{2R} h_l z^{R-l} = h_R \prod_{l=1}^{2R} \left(1 - \frac{c_l}{z}\right)$$

$$\sum_{l=0}^S a_l z^{S-l} = \prod_{l=1}^S \left(1 - \frac{d_l}{z}\right)$$

and so

$$\hat{h}(z) = h_R z^{S-R} \frac{\prod_{l=1}^{2R} \left(1 - \frac{c_l}{z}\right)}{\prod_{l=1}^S \left(1 - \frac{d_l}{z}\right)}$$