Ceng466 - Fundamentals of Image Processing Spring 2017-2018 Assignment 2

Duygu Dogan
e1941962@ceng.metu.edu.tr

***Index Terms*—order filters, median filtering, window size**

I. Introduction

This report aims to analyze implemented median filtering and come to a conclusion by a comparison with other filters and limitations and performance of median filter.

$$v(m,n) = median\{y(m-k, n-l), (k,l) \in W\} \quad (1)$$

II. Used Technologies

- Python3.6
- OpenCV

III. Implementation

Image below was used as source image.

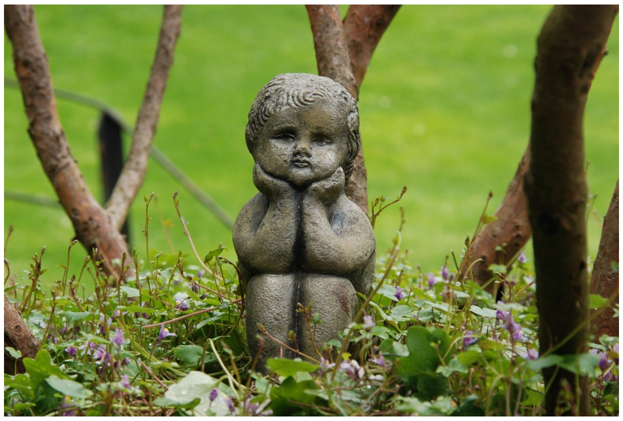


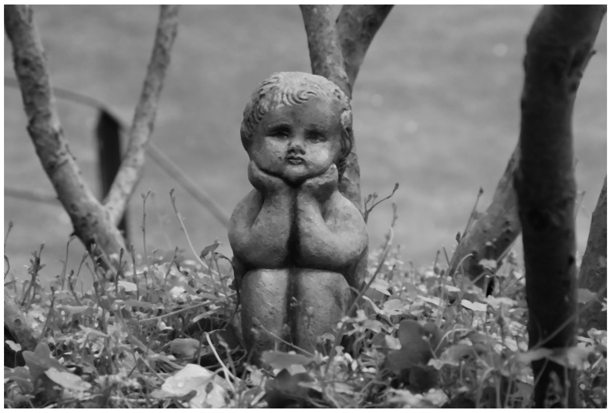
Fig. 1: Sample Colored Image.

In order to observe the performance of median filtering, steps below were followed:

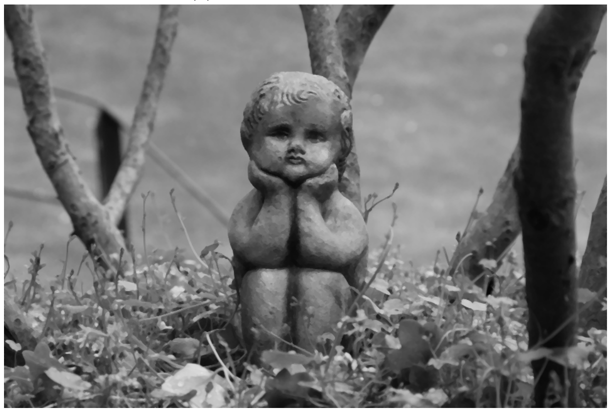
1) Colored image was converted to gray-scaled image.
2) Salt-and-pepper and Gaussian noise were separately added to the gray-scaled image.
3) 3x3 and 5x5 window sizes were used.

Limitations on Implementation:
Median Filtering could not reduce the noise on the border of the image. Thus, I assigned 0 values to those.

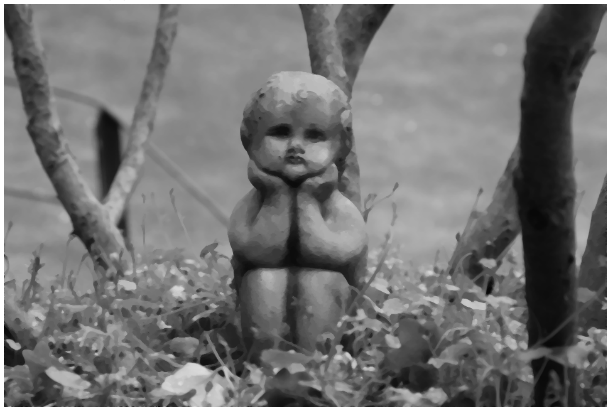
IV. Results



(a) 3x3 Median Filtered



(b) 3x3 Median Filtered Second Iteration



(c) 5x5 Median Filtered

Fig. 2: Median Filtering with Different Window Sizes

A. Parameter Comparison

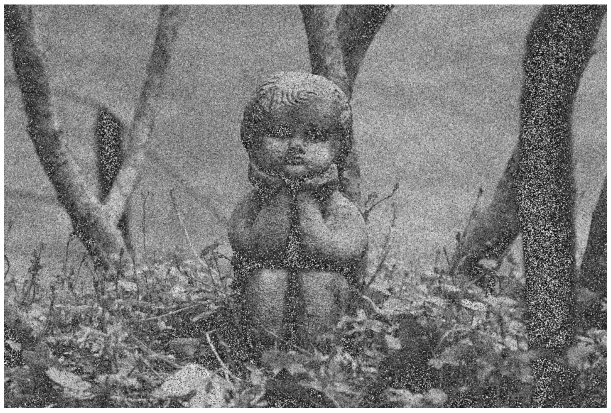
As can be seen from figures above, median filtering is useful for smoothing an image.

One of the important factors that effect its smoothing level is kernel size. In order to support this, 3x3 and 5x5 window sizes were applied to the image separately. Figure 3 clearly shows us that 5x5 kernel size makes an image smoother and less noisy yet more blurry.

To observe the differences between applying high order mask and iterating small order mask, 3x3 window sized median filtering was again applied to the already filtered image. By looking at the figure 2, we can clearly say that, using high order mask is much more effective (if the aim is to get smoother image) than iteration. As I interpreted this process, taking every 24 neighbor pixels is more effective than taking 8 neighbor pixels more than once.

B. Performance

It is useful for removing isolated pixels while preserving resolutions. Figures below show that the median filter performs very well on images containing impulse noise but performs poorly when the noise is Gaussian.


(a) Noisy Image

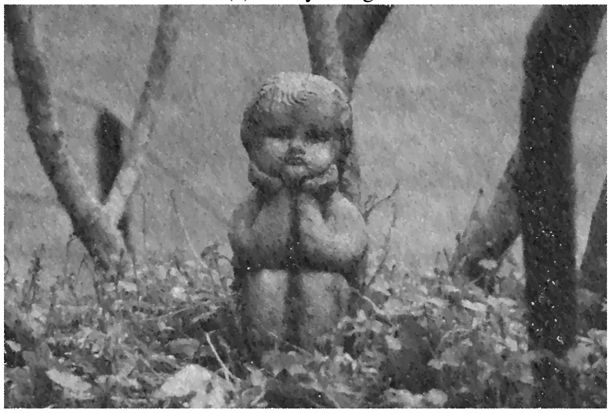
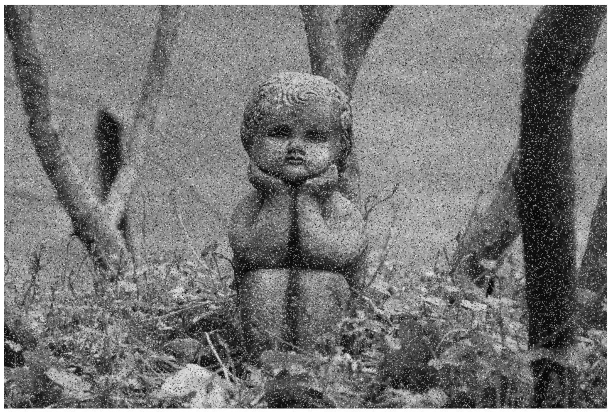

(b) Median Filtered Image

Fig. 3: Median Filtering on Gaussian Noise


(a) Noisy Image

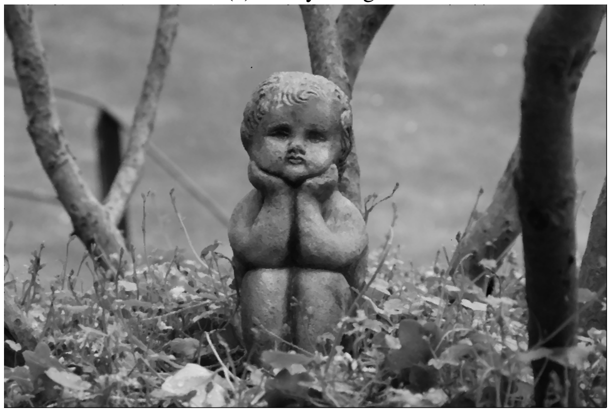

(b) Median Filtered Image

Fig. 4: Median Filtering on Salt and Pepper Noise

C. Limitation

It performs poorly when the number of noise pixels in the window is more than the half.

Additionally, sorting N pixels is high costly.

D. Comparison with Other Filters

Median filtering is highly effective in removing salt and pepper noise in comparison with the others. However, Gaussian filtering would be a better choice to remove Gaussian noise.

Yet, Arias-Castro et al. (2009) states that the median filtering is dramatically better at preserving edges than Gaussian and linear smoothing [1]. By looking at the figures above, we also can easily say that, the filtered images especially with a small window size have still conserves its edges comparing to the linear filtered images which were implemented on the previous assignment.

V. Notes

Salt-and-Pepper and Gaussian Noise was implemented based on the available codes.[2][3]

References

[1] E. Arias-Castro, & D. L. Donoho (2009), Does median filtering truly preserve edges better than linear filtering? The Annals of Statistics, 37(3), 1172-1206.

[2] K., Ingraham. (2017, February 4). Salt & Pepper Noise and Median Filters, Part II The Code. Retrieved from https://blog.kyleingraham.com/2017/02/04/salt-pepper-noise-and-median-filters-part-ii-the-code/

[3] Python code for Gaussian noise. (2017, May 12). Retrieved from http://www.magikcode.com/?p=240