International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 6 (2019), pp. 770-775
© International Research Publication House. http://www.irphouse.com

Comparison and Analysis of Noise Filtering Algorithms for Path Tracing

ByungGyoo Kim

Department of Computer Science, Keimyung University, 1095, Dalgubeol-daero, Dalseo-gu, Daegu, Republic of Korea.

ORCID: 0000-0003-1387-5108

SeongKi Kim

Department of Game mobile, Keimyung University, 1095, Dalgubeol-daero, Dalseo-gu, Daegu, Republic of Korea.

ORCID: 0000-0002-2664-3632

Abstract

Global illumination is an active research area in 3D computer graphics; it can add more realistic lighting effects to 3D scenes as it traces indirect light as well as direct light to render a scene. The path tracing method is the most popular approach for global illumination owing to its stability. However, it is still difficult to use path tracing for real-time rendering, despite the improvements in performance using general-purpose GPU (GPGPU), and data structures such as KD-Tree and bounding volume hierarchy (BVH). When path tracing is used, the shaded color becomes noisy in the case that the color is not yet converged, and this is one of the problems of path tracing. To eliminate this path tracing noise, a noise removal filter can be used. However, many filtering algorithms exist, and users have difficulty in selecting one of them. Here, we compare and analyze filtering algorithms such as high-pass filters (Sharpening filter) and low-pass filters (Mean, Median, and Gaussian filter) in terms of their performance and rendering quality. The comparison shows that the mean filtering algorithm has the best removal effect during the initial samplings while the median filtering algorithm has the best effect during the later samplings. However, the median filtering algorithm takes the longest time owing to its search for the median value.

Keywords - Filter Masking, Global illumination, High-Pass Filter, Image Filtering, Path Tracing, Pepper Noise

1. INTRODUCTION

Global illumination can render a scene more realistically compared to local illumination. It renders a scene after considering not only direct, but also indirect light. Among the many algorithms that can achieve global illumination, path tracing has been widely used because of its stability. However, the path tracing method uses the Monte Carlo technique and is very complex [1] as it requires the considering the effects of both direct and indirect light. In the Monte Carlo method, a solution is found by means of averaging all the intermediate results after sampling, rather than by calculating a large amount of indirect light one by one [2].

As path tracing uses the Monte Carlo method, it is inherently very complex and takes a long time to obtain the converged result. However, accelerated algorithms have been developed for real-time path tracing. KD-Tree and bounding volume hierarchy (BVH) are improved approaches that can reduce the computing time. General-purpose computation on the GPU (GPGPU) can also be used to accelerate the algorithm [3].

Despite these improvements, path tracing is still complex. A pixel becomes a noise when there are not enough samplings, or a pixel becomes a background color when it does not collide with any surfaces. This results in a large amount of pepper noises before enough samplings have been taken. Filtering algorithms such as mean, median, sharpening, and Gaussian filters can be used to mitigate these noises.

Here, we implemented the filtering algorithms, applied them to the results of path tracing, and then compared them in terms of rendering quality and performance. This paper is organized as follows. Section 2 describes the filtering algorithms that can be used for path tracing results; Section 3 describes the implementation of these algorithms; Section 4 compares them with respect to their performance and rendering quality; and Section 5 concludes this study.

2. RELATED WORKS

2.1 Pepper Noise at Path Tracing

Path tracing is one of the algorithms that can realize global illumination with a Monte-Carlo integration, which is performed by casting a ray to a random direction in the intersection case with an object. Path tracing comprises of the following processes: ray generation, intersection tests, and color shading. During ray generation, the algorithm generates a ray from a camera to every pixel on a plane. Then, it tries to find the nearest intersected 3D object after checking all the 3D objects. After finding the nearest object, the ray travels to a random direction in the collision case with diffuse material or is refracted or reflected to a different direction, when the collided with object has refraction or reflection properties. During color shading, the color of a pixel is determined according to the intersected 3D object. These processes constitute a single sample, and several samples are performed to render a single scene.

When we use path tracing for rendering a scene, a number of refractions and reflections need to be calculated to get the final value of a pixel. Theoretically, the path tracing can generate the correct color for a pixel if infinite time is given. However, infinite time cannot be given due to limited computational resources and time constraints; the path tracing stops bouncing a ray at a threshold depth or Russian roulette. As a result, much noise is rendered on the screen. The result is illustrated in Fig 1.

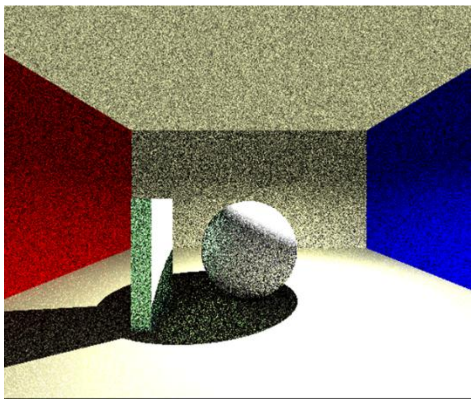


Figure 1. Path tracing before sufficient convergence

2.2 Mean Filter

A mean filter is a filter that calculates the mean value around a target pixel and replaces the target pixel with the filtered result. The advantage is that the noise of the entire image can be processed quickly when there are no significant differences among neighboring pixels. However, the mean filter gives the same weighting to all the neighboring pixels; thus, a pixel can be largely affected by surrounding pixels. Fig 2 shows the basic processes of mean filter.

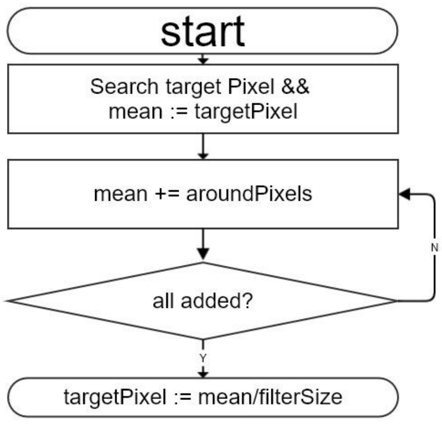


Figure 2. Mean filter

2.3 Gaussian Filter

A Gaussian filter is designed to reduce the effects of surrounding pixels, which is a disadvantage in a mean filter. The Gaussian filter weights neighboring pixels according to the normal distribution and decreases the weight as a pixel moves away from the target pixel. The Gaussian filter produces a clearer image than the Mean filter.

It generates an image between the original and mean filters and can maintain the original clarity to some degree. Fig 3 illustrates the basic processes of Gaussian filter.

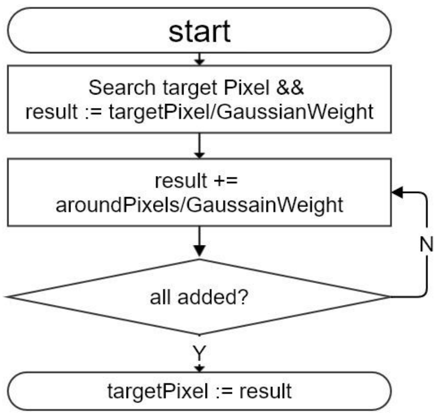


Figure 3. Gaussian filter

2.4 Median Filter

Mean and Gaussian filters tend to be vulnerable to salt and pepper noise [4]. A median filter examines the surrounding pixels of a target pixel, finds the median value, and replaces the value with the target pixel [5].

However, the Median filter is an algorithm that finds a middle value; thus, sorting is required, resulting in the requirement of additional time. The larger mask makes the performance slow. Fig 4 is the basic processes of median filter.

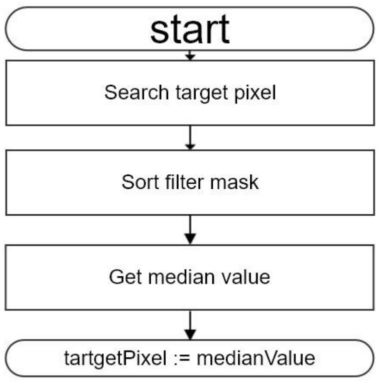


Figure 4. Median filter

2.5 Sharpening Filter

All the previously mentioned filters are low-pass filters, which are mainly used to remove noise or obtain blurred images. High-pass filters can be used to catch low-frequency signals and determine the outline (or boundary) of objects.

A Sharpening filter applies a high-pass filter to clearly represent the boundary line and noise. Fig 5 demonstrates the processes of sharpening filter.

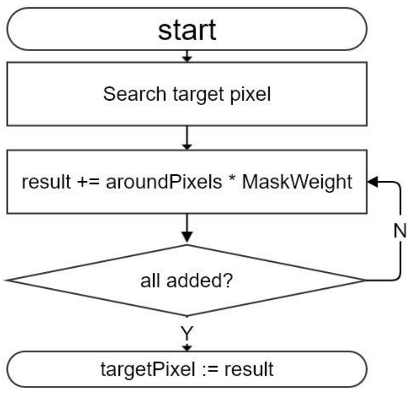


Figure 5. Sharpening filter

3. IMPLEMENTATION

3.1 Mean Filter

We implemented the mean filter, using a 3×3 mask, and applied it to the pixel position between width 1 and -1 and height 1 and – 1, to avoid the border area. Each neighboring color has a weighting of 1/9, as illustrated in Fig 6.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 6. Weights of neighboring pixels in Mean Filter

To implement the mean filter, we added all the neighborhood pixels and averaged them. We them replaced the result with a target value.

3.2 Gaussian Filter

The mask used for the Gaussian filter was derived using Eq. 1:

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

In Eq. 1, X is the value of a sample, Z is the estimate of the distribution, μ is the mean, and σ is the standard deviation.

The average value to be used in the image by applying Eq. 1 is zero; thus, assuming the weight of all pixels to be equal, the value obtained using the normal distribution can be weighted as illustrated in Fig 7.

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/9	1/16

Figure 7. Weights of neighboring pixels in Gaussian filter

3.3 Median Filter

To apply the Median filter, it is necessary to find the median value for the algorithm. However, to know the median value, a sorting is necessary. Thus, more time is required. Fig 8 illustrates our implementation of median filter.

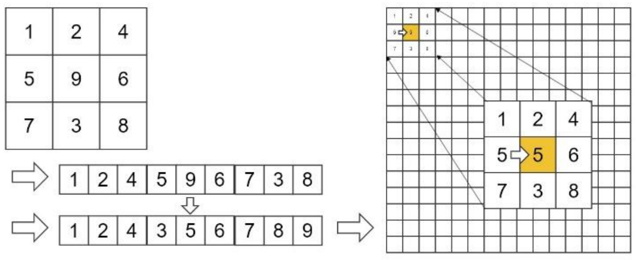


Figure 8. Median filter

3.4 Sharpening Filter

After examining the colors of all neighboring pixels, we measured the average weight of the surrounding pixels, and the weight of a target pixel. We then calculated the difference between those values. Noise and contour boundaries remained, which were then added to the original image to interpolate the values. After this, the original image was added, as illustrated in Fig 9; the noises were interpolated [6].

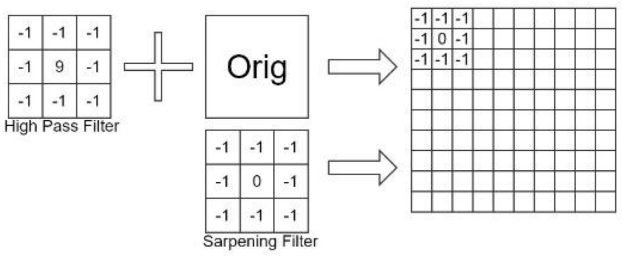


Figure 9. Sharpening filter

4. COMPARISON AND ANALYSIS

Here, we describe the results of the filtering algorithms.

4.1 Performance

For each filtering algorithm developed, the time taken to remove noise was measured 10 times and a mean time value was determined. The mean values for the different filter types are illustrated in Fig 10.

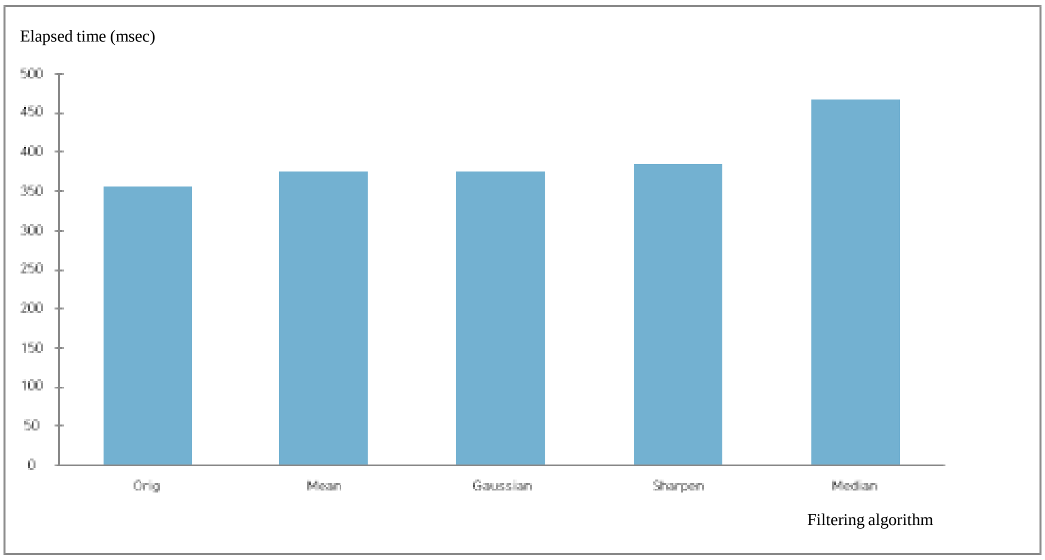
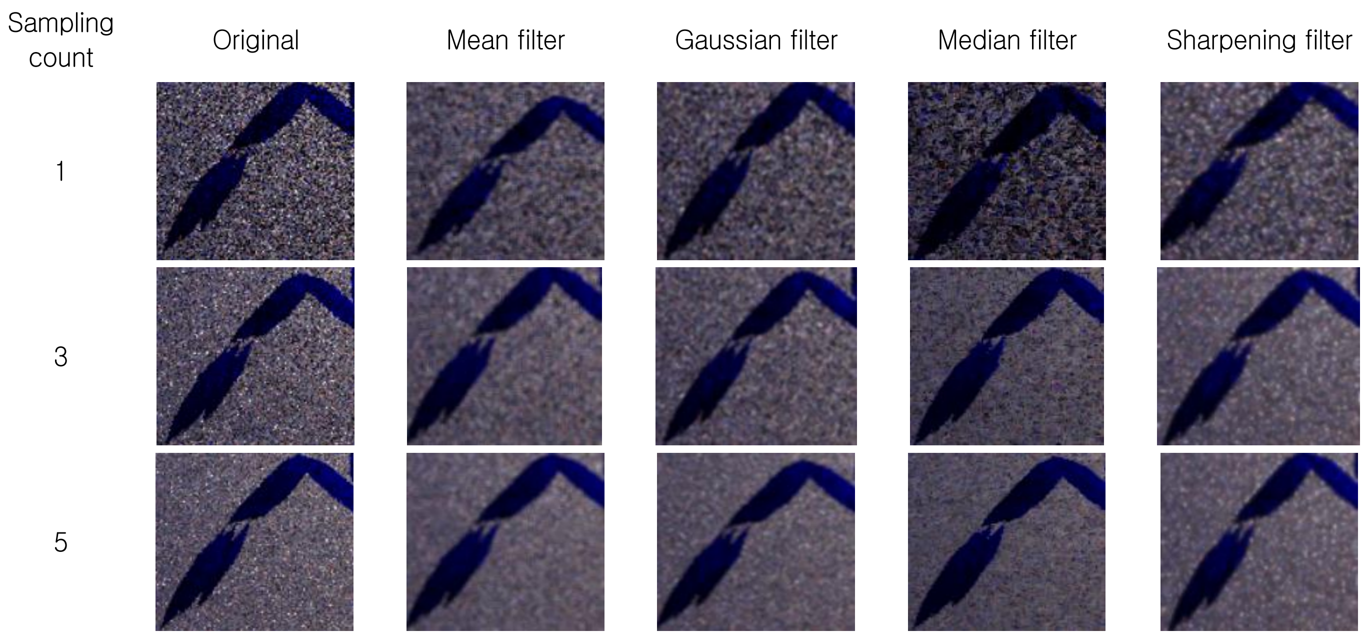


Figure 10. Elapsed time to draw single scene

In Fig 10, the performance of each filtering algorithm is illustrated. If no algorithms are used, the rendering takes 354 ms. When Mean and Gaussian filters are used, it takes 375 ms. Using a Sharpening filter takes 385 ms, and a Median filter takes 465 ms.

4.2 Rendering Quality.

Fig 11 shows the results of the filtering algorithms executed after path tracing. We executed the algorithms on Windows 10 (x64), Visual Studio 2017, Intel Core i5-6300HQ, and NVIDIA GeForce GTX960M. We rendered the ant model in a Cornell box [7], which has 486 vertices and 912 triangles. Based on this, the performance and rendering quality were compared. The comparison was made after 1, 3, 5, 10 and 20 samplings.



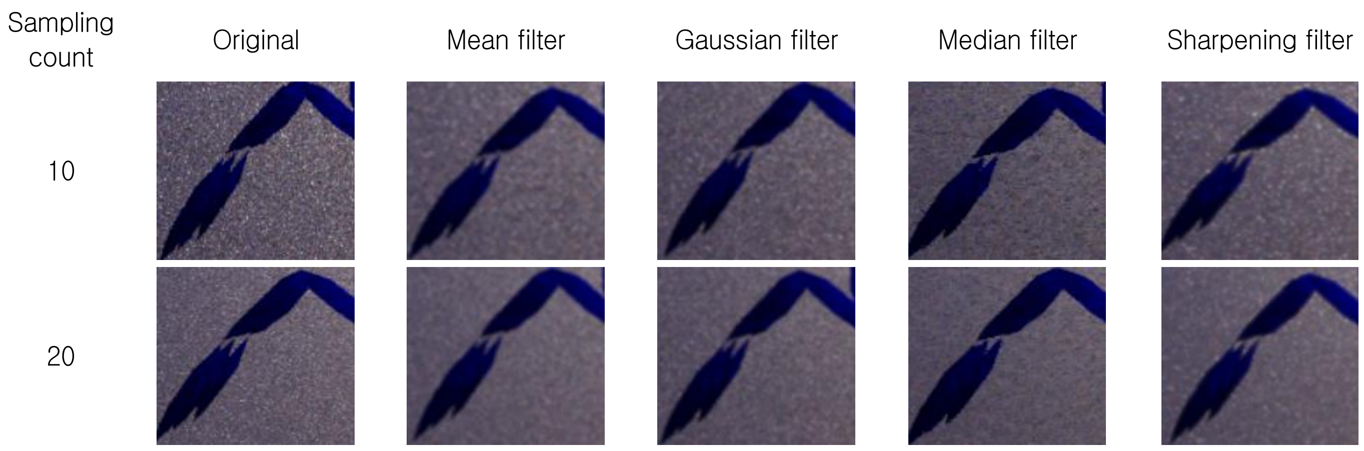


Figure 11. Filtering effects

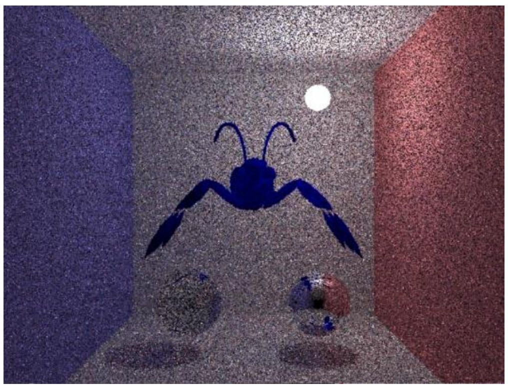


Figure 12. No filter (3 samplings)

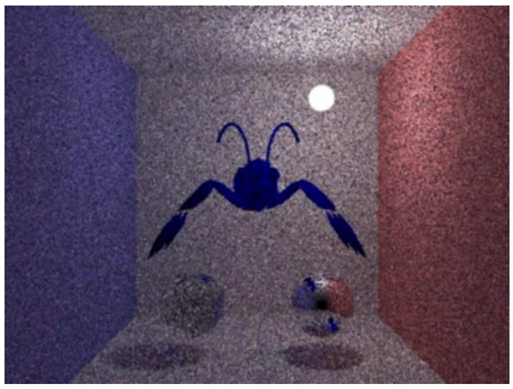


Figure 13. Mean filter (3 samplings)

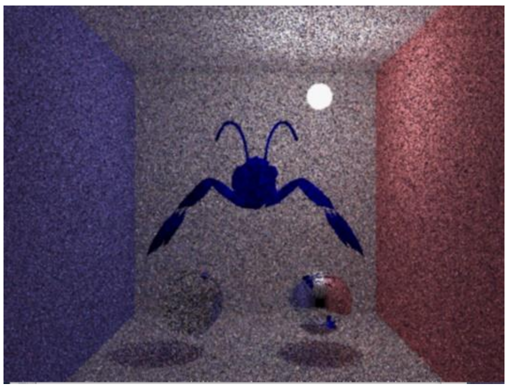


Figure 14. Gaussian filter (3 samplings)

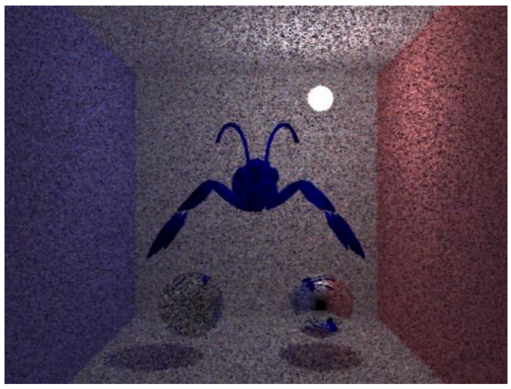


Figure 15. Median filter (3 samplings)

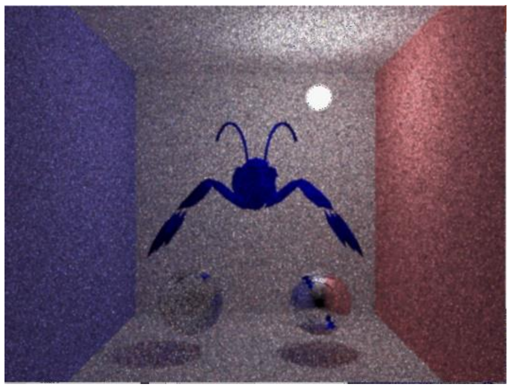


Figure 16. Sharpening filter (3 samplings)

As shown in Figs 12 and 16 above, the median filter most effectively decreases salt and pepper noise. The Mean filter makes the image smoother than the other filters, which is a fundamental characteristic of low-pass filters [8]. As shown in Fig 14, the Gaussian filter produced an image between the original image and the image rendered by the mean filter. As shown in Fig 16, the Sharpening filter decreases the sharpness.

5. CONCLUSION

The path tracing method can help satisfy the increasing demand for high-quality rendering; however, it is problematic in terms of noise before convergence through a sufficient number of samplings. To reduce noise, several filtering algorithms can be used. However, each algorithm has its own advantages and disadvantages. Thus, it is difficult to select one of them. To reduce this problem, we describe and implement filters, then compare them in terms of their performance and rendering quality. From this comparison, we conclude that the mean filter exhibits the best removal effect during the initial samplings, and the median filter exhibits the strongest noise removal effect during later samplings; however, the median filter requires more time. To the best of our knowledge, this study is the first that investigates the filters for the path tracing methodology. In future, we plan to study the interpolation method which uses artificial intelligence.

Acknowledgements

This research was supported by the National Research Foundation of Korea (NRF) (NRF-2017R1A1A1A05069806). SeongKi Kim is the corresponding author.

REFERENCES

[1] Lafortune, E.P., Willems, Y.D., 1993, "*BI-Directional Path Tracing*," Proceeding of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93), pp. 145–153, Alvar, Portugal.

[2] Spall, J.C., 2003, "*Estimation via Markov chain Monte Carlo*". IEEE Control Systems Magazine, *23(2)*: 34–45. doi:10.1109/MCS.2003.1188770.

[3] Kim, M., 2013, "*Kd-tree construction and optimization for real-time raytracing on the GPU,*" In Computer science.

[4] Ery, AC, Donoho, D.L., 2009, "*Does median filtering truly preserve edges better than linear fltering?,*" Annals of Statistics, *37(3)*: pp. 1172–1206.

[5] Veerakumar, T., Jayaraman, S., Esakkirajan. S., 2009, "*Digital Image Processing,*" Tata McGraw Hill Education, p. 272. ISBN 9781259081439.

[6] Mather, P.M., 2004, "*Computer processing of remotely sensed images: an introduction (3rd ed.),*" John Wiley and Sons, p. 181. ISBN 978-0-470-84919-4.

[7] Niedenthal, Simon, 2002 "*Learning from the Cornell Box,*" Leonardo, *35(3)*: pp. 249–254.

[8] Makandar, A., Halalli, B. 2015, "*Image Enhancement Techniques using Highpass and Lowpass Filters,*" International Journal of Computer Applications (0975–8887), *109(14).*