Improved Multi-scale Retinex For Image Enhancement Using Guided Filter And Customized Sigmoid Function With Its Implementation On FPGA

by

PARAS BHANWAL
202015006

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
ELECTRONICS AND COMMUNICATION

with specialization in
Wireless Communication and Embedded Systems
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

A program jointly offered with
C.R.RAO ADVANCED INSTITUTE OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE

May, 2022

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Electronics and Communications at Dhirubhai Ambani Institute of Information and Communication Technology & C.R.Rao Advanced Institute of Applied Mathematics, Statistics and Computer Science, and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Paras Bhanwal

Certificate

This is to certify that the thesis work entitled IMPROVED MULTI-SCALE RETINEX FOR IMAGE ENHANCEMENT USING GUIDED FILTER AND CUSTOMIZED SIGMOID FUNCTION WITH ITS IMPLEMENTATION ON FPGA has been carried out by PARAS BHANWAL for the degree of Master of Technology in Electronics and Communications at *Dhirubhai Ambani Institute of Information and Communication Technology & C.R.Rao Advanced Institute of Applied Mathematics, Statistics and Computer Science* under my/our supervision.

Dr. Yash Agrawal

Dr. Manish Khare

Acknowledgments

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisors, Dr. Yash Agrawal and Dr. Manish Khare, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of all my friends, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Contents

Abstract

Image enhancement is a technique used in digital image processing to remove or overcome the effects of noise, low illumination, blurriness, or color loss in the digital image. These effects arise during the process of image acquisition. Various other factors such as environmental conditions and data loss during image transmission can also affect the image quality. The presence of the these effects degrade the overall image quality. In application such as medical imaging, defence, aerial surveillance, traffic monitoring and others, where digital images are used for crucial purposes, it becomes very important to enhance the image before it can be used for the required purpose.

In low light environmental conditions when images are acquired by camera, poor contrast and color losses can be seen in several regions of the acquired image. To enhance the image under such conditions, researchers have proposed various techniques. Some techniques produce good contrast but lacks in color reproduction, while other produces good colors along with good contrast but intensify the noise present in the dark regions of the image.

In order to mitigate the issue of noise amplification while providing good color and contrast, we have proposed a retinex based image enhancement technique that uses a customized sigmoid function and guided filter for the image enhancement. We have compared the proposed method with the existing image enhancement methods on both qualitative and quantitative basis. For qualitative analysis we have tested the proposed method for multiple images, which are obtained under different environmental conditions and in different surroundings. For quantitative analysis we have used various image quality measures such as entropy, peak signal to noise ratio and others for comparison. The proposed technique provide good contrast in the areas affected by poor contrast and produce good colors in the same. The proposed method is capable of suppressing the enhancement of noise, hence showcasing its superiority with the compared techniques.
Keywords: *Guided filter, Image enhancement, Multi-scale retinex, Sigmoid function*

List of Tables

List of Figures

CHAPTER 1
Introduction

Human gather information from the surroundings through their five sensory organs which are ears, nose, eyes, tongue and skin. The information collected by each sensory organs is transferred to the human brain via nervous system. The visual information of the surroundings is gathered by the eyes and the collected information is then transferred to the visual cortex inside the brain via optical nerve. From this information the brain deduce certain knowledge which can be color, shape, and size of the objects present in the surroundings.

Digital cameras works on the same principle as that of human eye. The real world scene captured by the camera from its field of view is converted to digital data. The visual information acquired in form of digital data from the camera is then stored in a digital memory for later purposes. The process of capturing a real world scene with the help of a camera is called image acquisition. Image acquisition is the very first step of image processing. Each point in real world scene has a certain color and intensity level associated to it. In order to store these information in the memory, a color scheme is used. For every scene in real world, the digital data generated by the camera is of size $M \times N \times C$. Here $M \times N$ represents

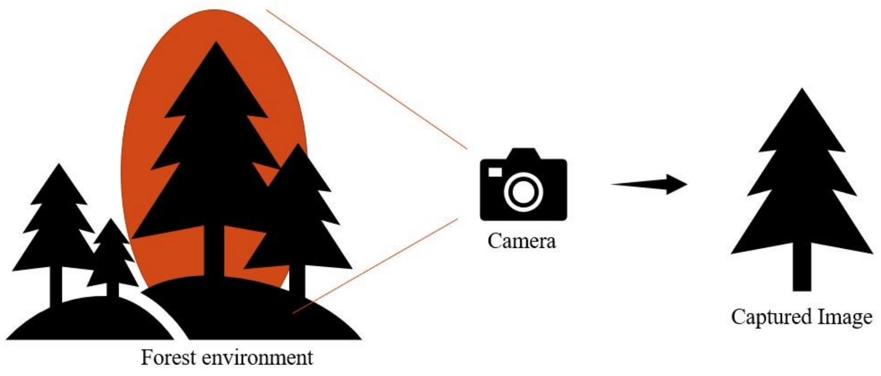


Figure 1.1: Image capturing with a camera

total number of pixels present in the digital image and C represents color channels available for each pixel. The number of intensity level that can be reproduced by each pixel depends upon the size of each data element usually an 8-bit data is used for this purpose. Multiple color schemes are used in image processing some of them are as follows:

a) Gray-scale - In this color space only the intensity information of each pixel is stored. This color space is most primitive of all color spaces.

b) RGB - In this color space, both the intensity and color of the pixel are stored. Every color in this color space is represented as a combination using these three color Red, Green and Blue. Various application of the RGB color space is in web graphics, image display in computers and television. For this color scheme $C = 3$.

c) CMY - This color space is also known as subtractive color model. Unlike RGB color space in CMY color space every color is represented as combination of cyan, magenta and yellow color. The equation 1.1 shows color space conversion between RGB and CMY. The CMY color space is used in commercial printing such as books and magazine. Even for this color scheme $C = 3$. The color space conversion between RGB and CMY is shown below:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1.1}$$

d) HSV - This color model is more close in representing how human eye perceive different color. This color space is more complex then other color spaces. The Hue component represent the purity of color. Saturation component shows the degree with which the color is mixed with white color. The Value component is used to store the brightness level of the color. The color space conversion between RGB and HSV is shown below:

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$C_{max} = max(R', G', B')$$

$$C_{min} = min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

$$H = \begin{Bmatrix} 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{Bmatrix} \tag{1.2}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

$$V = C_{max}$$

Depending upon camera specifications and environmental surrounding conditions there may arises multiple issues during the image acquisition process which leads to degradation in quality of the captured image. The camera specification that affects the captured image is its resolution, which relates to total no of pixel generated per image. A lower resolution of the captured image generally means that multiple points in the real world scene are mapped to a single pixel in the image, which leads to visual information loss. In order to store more visual information camera with higher resolution is used. During low light environmental conditions the noise generated from the electrical circuit inside the camera might get added onto the dark or low illumination regions in the real world scene. This noise will make these regions appear unnatural. Other environmental conditions can also affects the captured image quality such as fog and rain. Improper handling of camera can also introduce blurriness in the captured image which is not desired in practical applications.

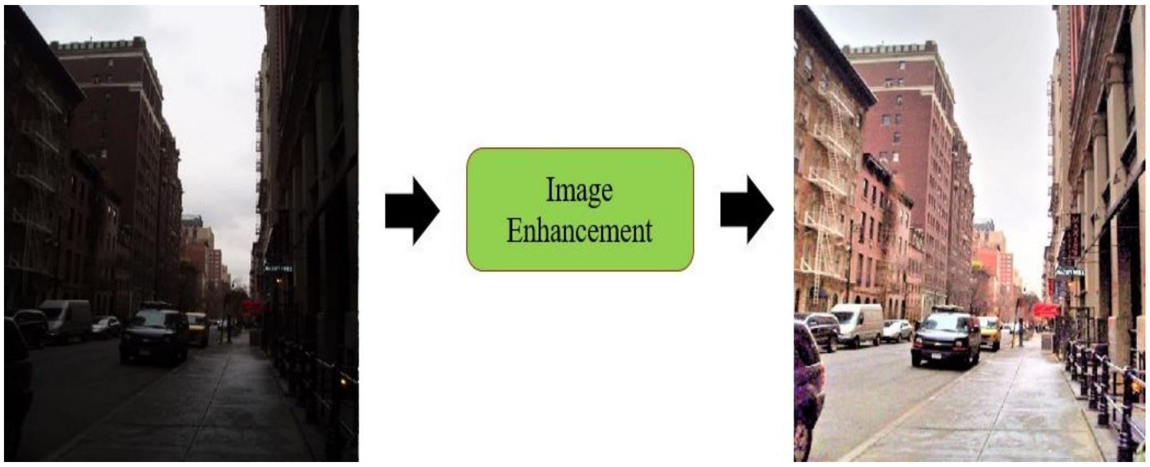


Image credit: Image from LIME dataset
Figure 1.2: Effects of image enhancement on a low quality image

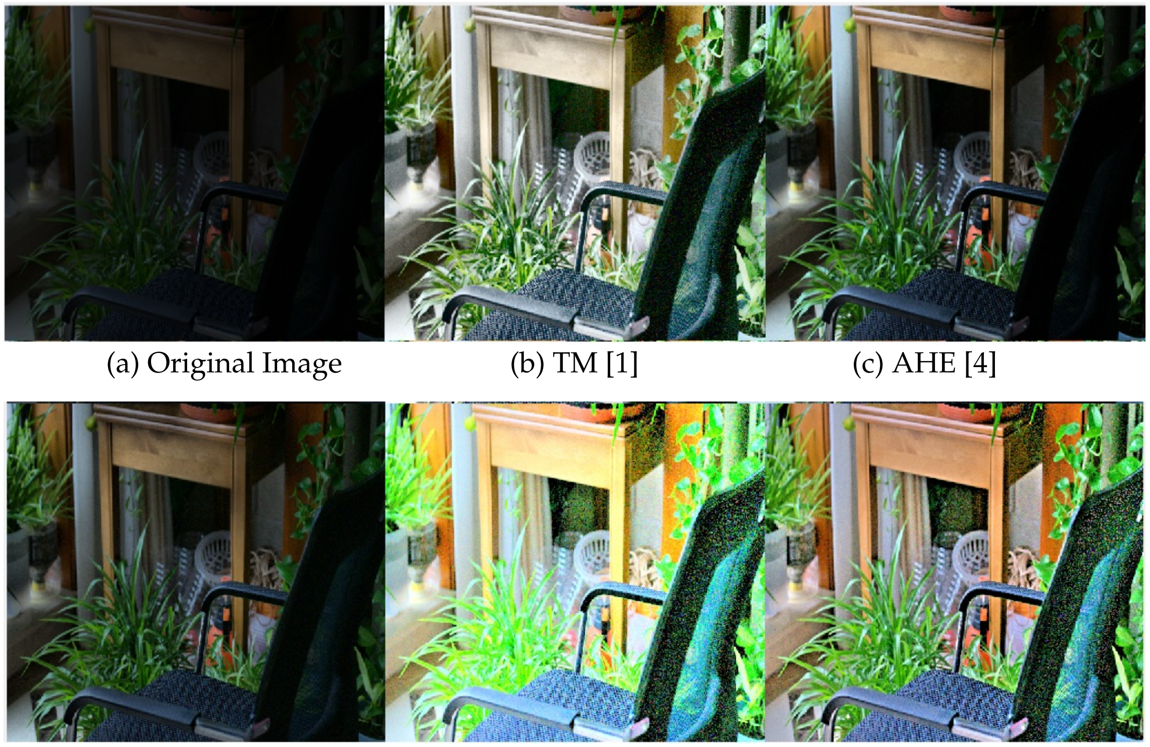
To mitigate these effects, image enhancement technique is applied onto the degraded image. Depending upon the effect to remove various image enhancement technique exists. For noise removal two types of technique exists one is spatial domain based and the other is frequency. In spatial domain to remove noise, various filters are used such as mean, median, Wiener and Bilateral. If the image is suffering from multiple noises then a combination of these filter can be used. In spatial domain filter is applied directly onto the image data.

In frequency domain the image is first converted from spatial domain to its frequency domain using transformation techniques such as DFT, DCT, and DWT etc. Then after various frequency domain filters can be applied to remove the noise such as high pass, low pass, and wavelet based filter. The frequency domain noise removal techniques are more robust when compared to spatial domain based noise removal techniques as removing noise from frequency domain is much easier than removing it from spatial domain.

Contrast of a image relates to how well two different object can be distinguished from each other. A higher contrast of the digital image would relate to proper identification of different objects in the image. In digital image lower contrast issue arises due to poor lightning conditions during acquisition of the image. To increase the overall contrast mainly three types of techniques are used. These are Global contrast improvement techniques, local contrast improvements and hybrid contrast improvement techniques. Global contrast improvements techniques such as tone mapping (TM) [1], histogram equalization (HE) [2] and gamma correction [3] are fast and only introduce global contrast improvements while the improvements in local levels are unattended. Local contrast improvements techniques such as AHE [4] , AGCWD [5] do better improvements at local levels, these improvements often leads to uneven contrast improvements regionally as only the small regions in the image are attended. Hybrid contrast improvements techniques combine both the advantages of local and global contrast improvements techniques. Hence keeping a good balance between global and local contrast improvements.

1.1 Motivation

Retinex theory based techniques [6], [7], [8], [9], [10], [11] when compared with non retinex based techniques has a very clear advantage in terms of good colors reproduction and contrast in the enhanced image. A comparison between few no-


(a) Original Image (b) TM [1] (c) AHE [4]

(d) AGCWD [5] (e) MSRCP [8] (f) MSRGF [9]

Image credit: Image from LIME dataset

Figure 1.3: A comparison between non-retinex and retinex based image enhancement techniques

retinex based image enhancement techniques and retinex based techniques can be seen in Figure 1.3. The superiority of retinex based techniques makes them a suitable candidate to be used in various practical application such as medical imaging, under water imaging, image dehazing, satellite imaging and others. Researchers in [12] proposed a multi-scale retinex based approach for endoscopic vision augmentation used in robotic surgery, where other researcher in [13] proposed an under image enhancement algorithm for under water imaging. An image dehazing method was proposed by researchers in [14] which uses retinex theory as there base, similarly researchers in [15] proposed an image enhancement algorithm based on retinex theory for contrast improvements in remote satellite imaging.

Motivated from such works we have also picked up the mantle to improvise and enhance previously designed retinex based methods for image enhancement.

1.2 Objective

The primary objective of our work is to provide a more robust image enhancement technique that along with providing good colors and contrast is capable of noise suppression. For this we will studying various state of the art retinex theory based image enhancement techniques.

FPGA implementation has to be carried for the proposed technique in order to showcase its feasibility for use in various practical applications.

1.3 Organisation of thesis

The thesis organization is as follows:

Chapter 2) In this chapter state of the art retinex based image enhancement techniques are discussed along with their implementation benefits and drawbacks.

Chapter 3) In this chapter the proposed method and its flow of implementation are discussed.

Chapter 4) Ongoing FPGA implementation of the proposed method is discussed in this chapter.

Chapter 5) Results and analysis of the proposed method in comparison with methods discussed in Chapter 2 are discussed in this chapter.

Chapter 6) Results and analysis of the hardware implementation is discussed in this chapter.

Chapter 7) This chapter provides the overall Conclusion of the thesis.

CHAPTER 2
Related Works

In this chapter image enhancement techniques that are being used for contrast enhancement and color restoration in the filed of image processing are discussed.

2.1 Retinex Theory

Retinex theory is based on how the human eye perceives different colors. It was first proposed by Land *et al.* [6]. Retinex theory uses illumination component *L(x,y)* and reflectance component *R(x,y)* of the image for enhancement. The image captured *S(x,y)* by the camera can be represented as product of the illumination and reflectance component of the image. The illumination component represents the amount of light falling onto the surface of the real world scene from its surroundings, while the reflectance component represents the ability of the surface to reflect light from its surface towards camera. It is the reflectance component that contains useful information, mainly the color information of the object. Hence if we can extract the reflectance component of the input image, we can quickly identify areas affected by poor contrast and color loss in the image.

$$S(x,y) = L(x,y) \times R(x,y) \tag{2.1}$$

The illumination component is approximated using convolution of the captured image with a Gaussian kernel $G(x,y,c)$,

$$L(x,y) = S(x,y) * G(x,y,c) \tag{2.2}$$

Here the Gaussian kernel $G(x,y,c)$ is defined as:

$$G\left(x,y,c\right) = K \times e^{\left(-\frac{x^2+y^2}{c^2}\right)} \tag{2.3}$$

Where K is the normalization factor such that sum of all values in *G(x,y,c)*

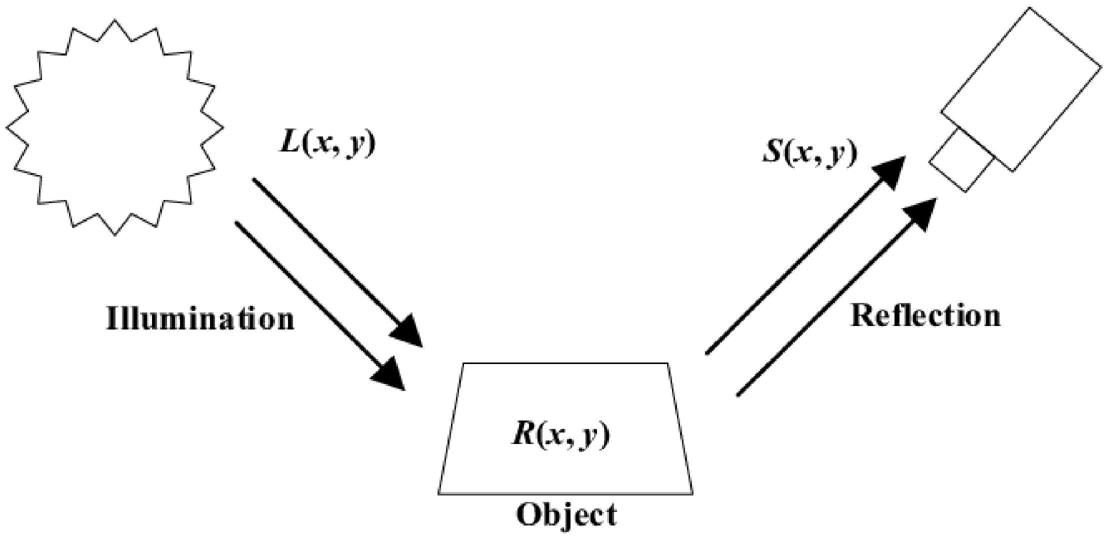


Figure 2.1: An illustration of retinex theory

becomes 1 and c is called the standard deviation or scale of the Gaussian kernel.

Now the reflectance component can be extracted from above equations using the log operator

$$\log R(x,y) = \log S(x,y) - \log[S(x,y) * G(x,y,c)] \tag{2.4}$$

The above equation uses only one Gaussian kernel to compute the reflectance component, hence it is called the single-scale retinex (SSR) algorithm. The above equation produces data values that can be positive or negative so in order to display the results in image domain linear stretching is applied onto $\log R(x,y)$. After linear stretching the data will be in image range [0,255].

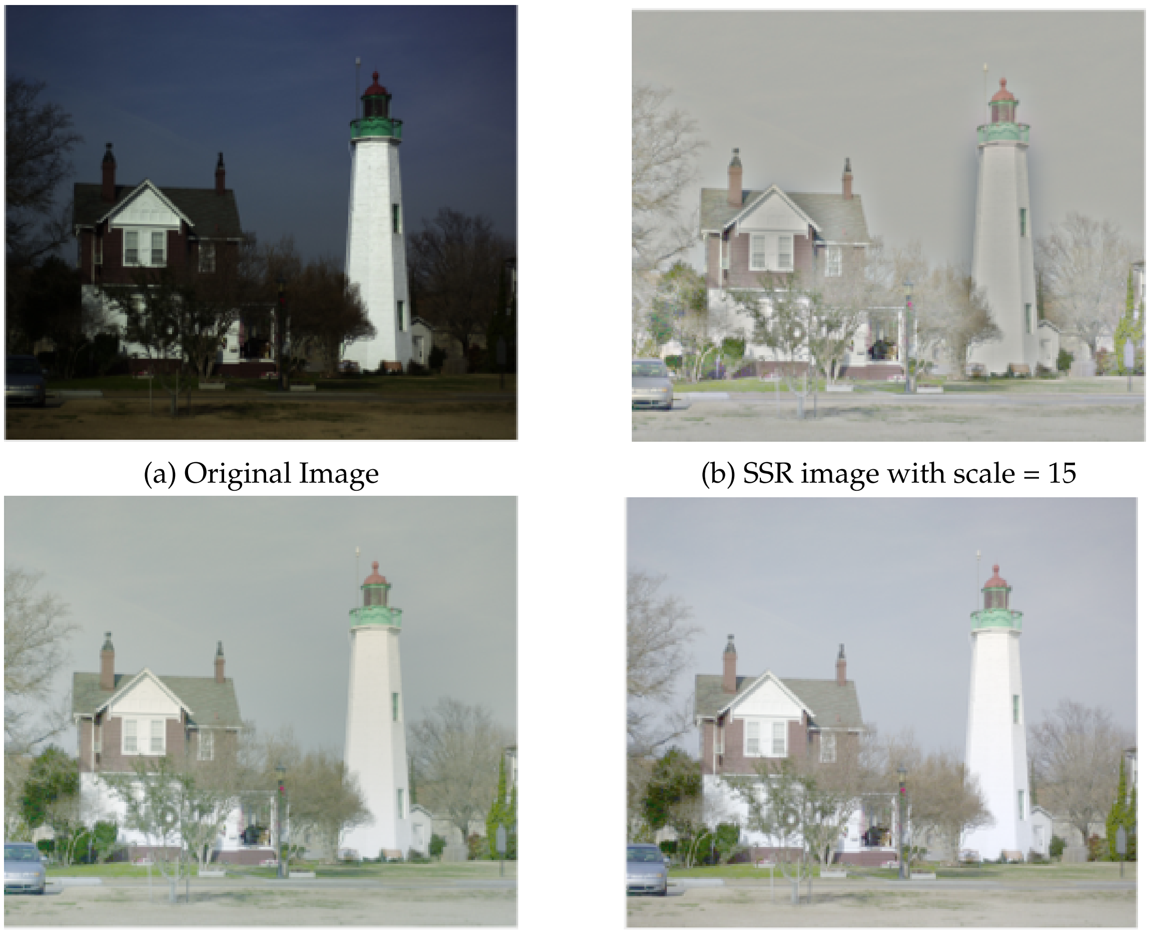
$$R(x,y) = 255 \times \left(\frac{\log R(x,y) - i_{min}}{i_{max} - i_{min}} \right) \tag{2.5}$$

here $i_{min} = min_{(x,y)}(\log R(x,y))$ denotes the min pixel value of $\log R(x,y)$, and $i_{max} = max_{(x,y)}(\log R(x,y))$ denotes the max pixel value of $\log R(x,y)$.

2.1.1 Issues with SSR image enhancement technique

The various issues associated with SSR are discussed as follows:

a) For a colorful image all color channel has to be processed with SSR. when using high-scale Gaussian kernel the time consumed for image enhancement


(a) Original Image (b) SSR image with scale = 15

(c) SSR image with scale = 80 (d) SSR image with scale = 250

Image credit: http://dragon.larc.nasa.gov/retinex/
Figure 2.2: Effects of using multiple Gaussian scales for SSR

will increase significantly.

b) When using small-scale Gaussian, the single-scale retinex algorithm provide good dynamic range compression (dynamic range compression relates to how well we can represent the original pixel value range of an image into a smaller range of pixel values). Beside providing good dynamic range compression SSR also introduce "haloing" artifacts for small scale of the Gaussian kernel.

c) Use of high-scale Gaussian in retinex algorithm provides more tonal rendition than small-scale Gaussian (tonal rendition relates how good colors are produced in the enhanced image). But the use of High scale Gaussian compromises on the details (sharp edges) of the image that can be seen in Figure 2.2.

d) The SSR cannot achieve overall good quality of image as finding one optimal

Gaussian kernel for different types of images that balances both dynamic range compression and color tone is difficult.

e) After processing is also required to enhance the colors and overall contrast in the SSR output image.

f) Noise present in extreme low light regions of an image is enhanced by the SSR.

2.2 Multi-scale Retinex (MSR)

In order to provide a solution for balancing dynamic range and tonal rendition, multi-scale retinex method was proposed by Jobson *et al.* [7] The MSR uses not one but three different scale Gaussian kernel for the image enhancement. Using these kernels, the reflectance component is computed, which is given as:

$$\log R_n\left(x,y\right) = \log I\left(x,y\right) - \log\left[G\left(x,y,c_n\right) * I\left(x,y\right)\right] \tag{2.6}$$

linear stretching is applied to every reflectance component obtained from above equation.

$$R(x,y) = 255 \times \left(\frac{\log R_n(x,y) - i_{min}}{i_{max} - i_{min}}\right) \tag{2.7}$$

here $i_{min} = min_{(x,y)}\left(\log R_n\left(x,y\right)\right)$ denotes the min pixel value of $\log R_n\left(x,y\right)$, and $i_{max} = max_{(x,y)}\left(\log R_n\left(x,y\right)\right)$ denotes the max pixel value of $\log R_n\left(x,y\right)$.

Hence the MSR reflectance component R_{MSR} is given as:

$$R_{MSR} = \sum_{n=1}^{N} w_n R_n(x,y) \tag{2.8}$$

Here w_n is the weigh value associated with each single-scale retinex $R_n(x,y)$ such that sum of all weigh values from w_1 to w_N is equal to 1. Here N represents total no of different scales used for MSR. As proposed by Jobson using three different scales and identical weigh values works well for most images. The different scale values are chosen in such a way that they keep the balance between details (high-frequency component of the image) and fidelity (low-frequency component of the image). A low scale Gaussian kernel will provide the dynamic range compression while a higher-scale Gaussian kernel will enhance the color rendition and a medium scale Gaussian kernel for balancing both.

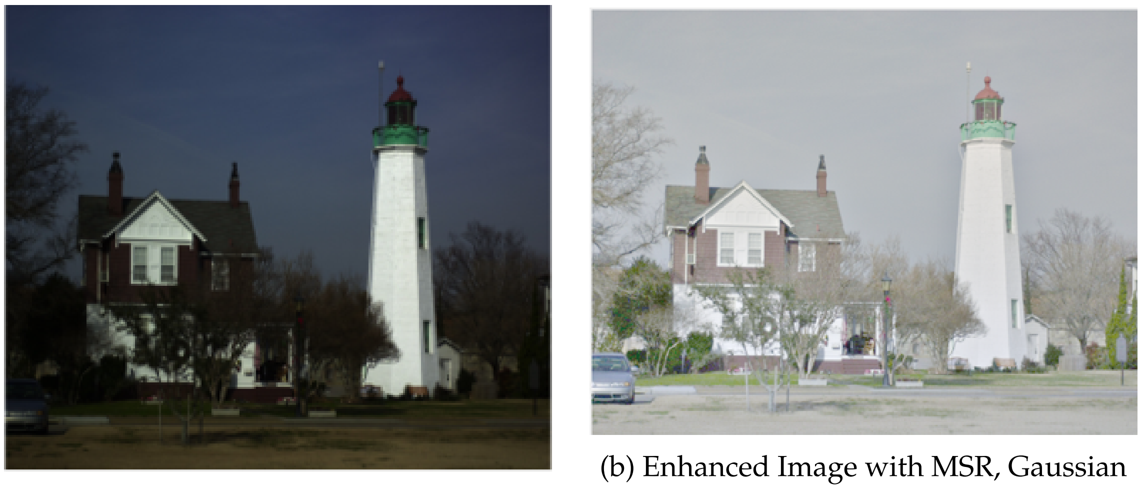
2.2.1 Issues with MSR image enhancement technique

The issues associated with MSR are discussed below:

a) For a color image each color channel has to be processed with MSR. Time consumption of MSR is very high as three color channels has to processed with three different size Gaussian kernel. MSR is around 3 times slower than SSR.

b) Similar to SSR, MSR also require some post processing to improve the overall color and contrast in image which can be observed in Figure 2.3.

c) MSR also suffers from noise amplification in extreme low light regions of image.



(a) Original Image

(b) Enhanced Image with MSR, Gaussian scales used c_1=15, c_2=80 and c_3=250

Image credit: http://dragon.larc.nasa.gov/retinex/
Figure 2.3: Image enhancement using MSR

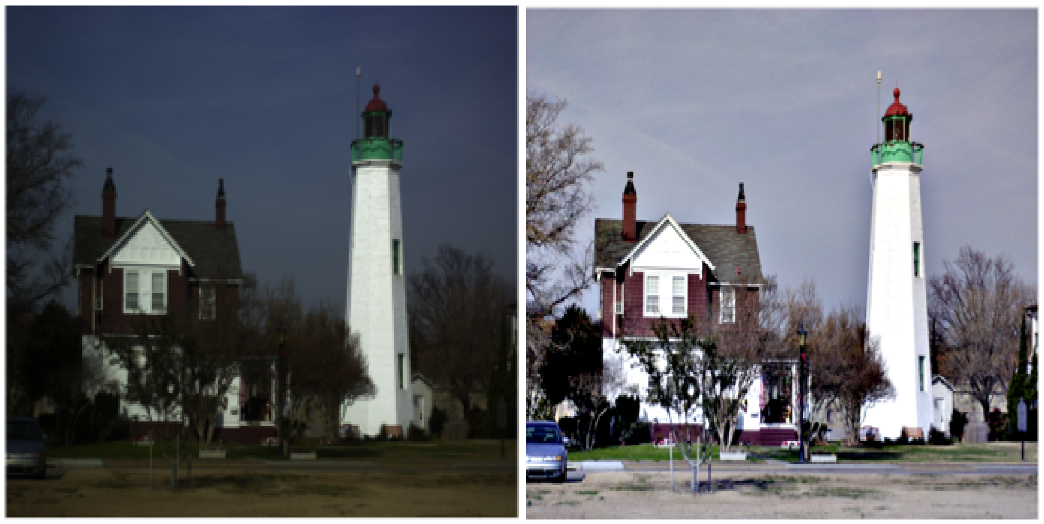
2.3 Multi-scale Retinex with Color Restoration (MSRCR)

To improve the lower color and contrast issue associated with the MSR method, multi-scale retinex with color Restoration was proposed by Jobson *et al.* [8]. In this method a color restoration function is applied just after the MSR processed image. The color restoration function used by jobson in their works is:

$$C_i(x,y) = \beta \log \left[\frac{\alpha S_i(x,y)}{\sum_{i=1}^{N} S_i(x,y)} \right] \tag{2.9}$$

here i the color channel under processing and N represents total number of color channel in image. and the final MSRCR enhanced image is given by

$$R_{MSRCR_i}(x,y) = C_i(x,y)R_{MSR_i} \tag{2.10}$$



(a) Original Image (b) Enhanced Image with MSRCR

Image credit: http://dragon.larc.nasa.gov/retinex/
Figure 2.4: Image enhancement using MSRCR

2.3.1 Issues with MSRCR image enhancement technique

The issues associated with MSRCR are discussed below:

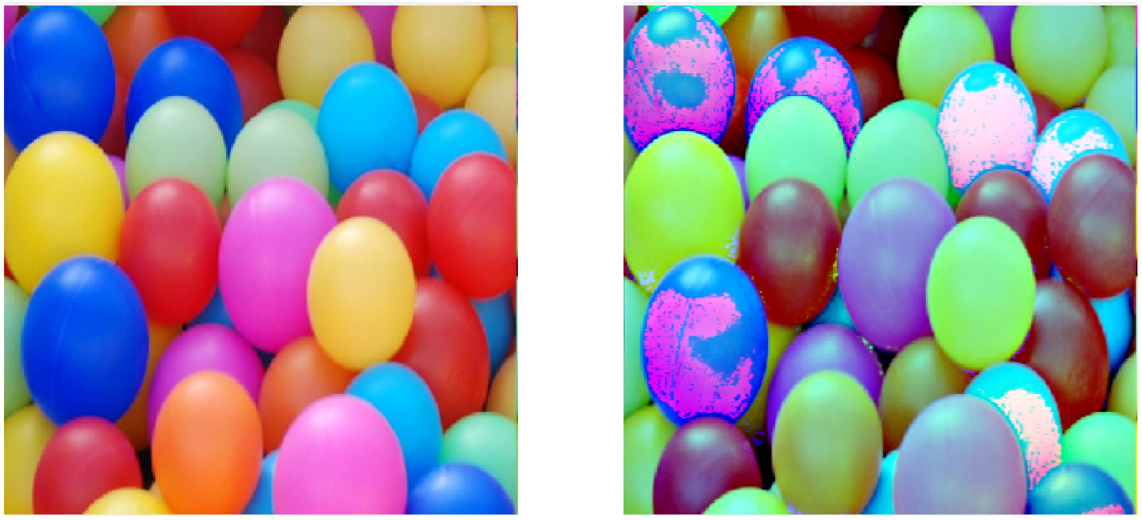
a) MSRCR does much better than MSR in terms of color and contrast improvements, but in certain cases the overall contrast and colors in image are still dull.

b) Time consumption is on similar level as that of MSR.

c) In certain images that contains big colorful regions the image introduce color artifacts as seen in Figure 2.5

2.4 Multi-scale retinex with chromacity preservation (MSRCP)

The issue of dull color in MSRCR was resolved in the method MSRCP proposed by Petro *et al.* [9]. A pseudo-code for MSRCP is shown in Figure 2.6 below:



(a) Original Image (b) Enhanced Image with MSRCR

Image credit: J.L. Lisani CC BY

Figure 2.5: Issues with image enhancement using MSRCR

```
Data: $I$ input color image; $\sigma_1, \sigma_2, \sigma_3$ the scales; $s_1, s_2$ the percentage of clipping pixels on each
      side
Result: MSRCP output color image
begin
    $\mathrm{Int} = (I_R + I_G + I_B)/3$                          ▷ Compute the intensity channel
    foreach $\sigma_i$ do                                          ▷ For each scale
        $\mathrm{Diff}_i = \log(\mathrm{Int}) - \log(\mathrm{Int} * G_{\sigma_i})$   ▷ Single Scale Retinex
    end
    $\mathrm{MSR} = \sum_i \frac{1}{3}\mathrm{Diff}_i$                  ▷ MultiScale Retinex
    $\mathrm{Int}_1 = \mathrm{SimplestColorBalance}(\mathrm{MSR}, s_1, s_2)$
    foreach pixel $i$ do
        $B = \max(I_R[i], I_G[i], I_B[i])$
        $A = \min\left(\frac{255}{B}, \frac{\mathrm{Int}_1[i]}{\mathrm{Int}[i]}\right)$   ▷ Compute the amplification factor
        $\mathrm{MSRCP}_R[i] = A \cdot I_R[i]$                    ▷ Compute each color channel
        $\mathrm{MSRCP}_G[i] = A \cdot I_G[i]$
        $\mathrm{MSRCP}_B[i] = A \cdot I_B[i]$
    end
end
```

Figure 2.6: Pseudo-code for MSRCP algorithm

As seen form Figure 2.7 the MSRCP method provide good color when compared to MSRCR, This method is also faster than MSRCR as in MSRCP only one channel (illumination) is processed with MSR.

2.4.1 Issues with MSRCP image enhancement technique

The issues associated with MSRCP are discussed below:

a) MSRCP does not work well with images in low light as it tends to enhance

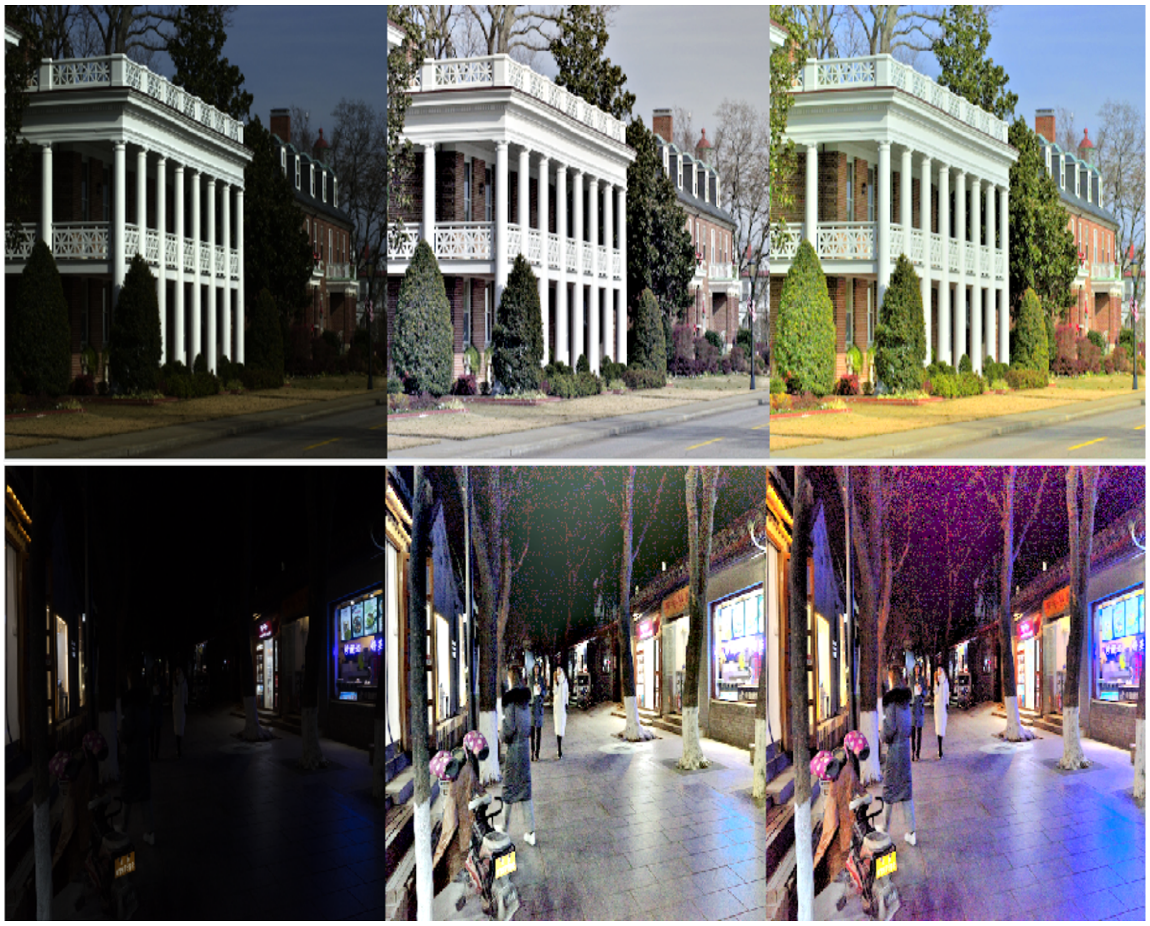

(a) Original Image (b) MSRCR [8] (c) MSRCP [9]

Image credit: Top image - http://dragon.larc.nasa.gov/retinex/
Bottom image - Image taken from Dark Face Dataset
Figure 2.7: comparison between MSRCR and MSRCP

noise present in the extreme dark region of the image as seen in Figure 2.7

b) In some situations pixel intensity in the enhanced image is unnatural.

c) Detail losses in small regions of image due to Gaussian kernel size.

2.5 Multi-scale retinex with guided filter (MSRGF)

The issue of detail loss due to use of Gaussian kernel size in previously discussed multi-scale retinex based methods were solved in the method proposed by Tang *et al.* [10]. The guided filter preserves details better than Gaussian filter, while filtering out noise. A detailed study of guided filter is discussed in next section.

2.5.1 Guided filter

Guided filter is a better edge preserving filter than Gaussian filter, the guided filter along with preserving details is also capable of removing noise. Steps included for guided filter is discussed below: The input image is represented as p, the output image q is given by a linear transformation between guidance image I in a square window ω_k with radius r centered at pixel k

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \tag{2.11}$$

The boundary of an item is related to its gradient. local linear model ensures q only has an edge only if $\nabla q = a\nabla I$. to obtain the coefficients we can use a cost minimization function:

$$E(a_k, b_k) = \sum_{i\epsilon\omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2) \tag{2.12}$$

In the above formula ϵ is a regularization parameter penalizing large a_k. the solution to the cost function is given by

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\epsilon\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon} \tag{2.13}$$

$$b_k = \bar{p}_k - a_k \mu_k \tag{2.14}$$

here μ_k and σ_k^2 are the mean and variance of I in ω_k; $|\omega|$ is the number of pixels in ω_k; $\bar{p}_k = \frac{1}{|\omega|}\sum_{i\epsilon\omega_k} p_i$ is the mean of p in ω_k. then the filtered output image can be computed using equation 2.11.

2.5.2 MSRGF algorithm

A detailed flow of the MSRGF method is shown in Figure 2.8. The input image is first converted to HSI color domain. In this method MSR processing is applied onto the intensity color channel I_v.

Using the guided filter filtered image can be obtained as follows:

$$L_i = GuidedFilter(I_v, r_i, \epsilon_k) \tag{2.15}$$

The first filtered image using guided filter is given as L_1, L_2 is the second filtered image. High frequency information is extracted using different guided filter scales from following

$$A_1 = \log(I_v) - \log(L_1) \tag{2.16}$$

$$A_2 = \log(I_v) - \log(L_2) \tag{2.17}$$

below equation is used to extract low frequency details :

$$LL = \log(I_v) - \log(A_1) \tag{2.18}$$

High frequency and low frequency details are then combined to produce the enhanced image.

$$e = \lambda_1 * A_1 + \lambda_2 * A_2 + LL \tag{2.19}$$

contrast in the image e is then enhanced by

$$min = mean(e) - 2.1 * var(e) \tag{2.20}$$

$$final = 255 \times \left(\frac{e - min}{\gamma \times var(e)} \right) \tag{2.21}$$

where $mean(e)$ is the mean of image e and $var(e)$ is the standard deviation of image e. The MSRGF enhanced intensity channel final is then combined with Hue (H) and saturation (S) component, the enhanced HSI color image is then converted back to RGB color space. The MSRGF produce good color in enhanced image while preserving good details this method can be very useful in indoor imaging applications.

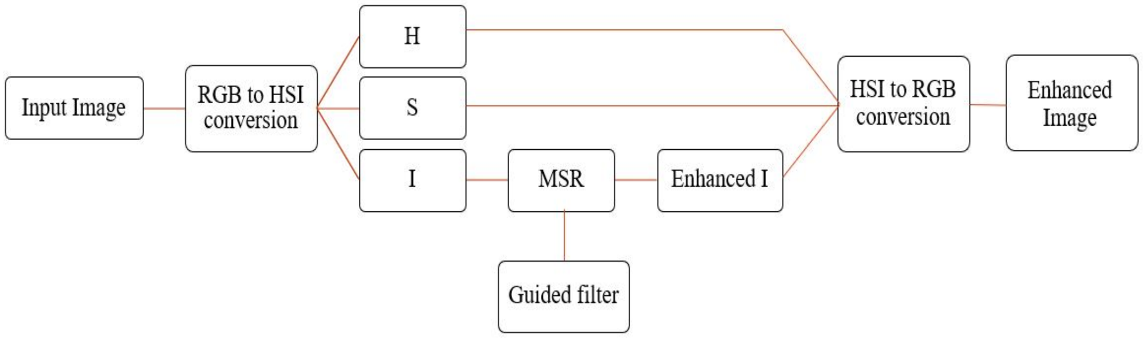


Figure 2.8: Flow chart for MSRGF algorithm

2.5.3 Issues with MSRGF image enhancement

The issues associated with MSRGF are discussed below:

a) The amplification of noise in low light regions of image is still an persistent

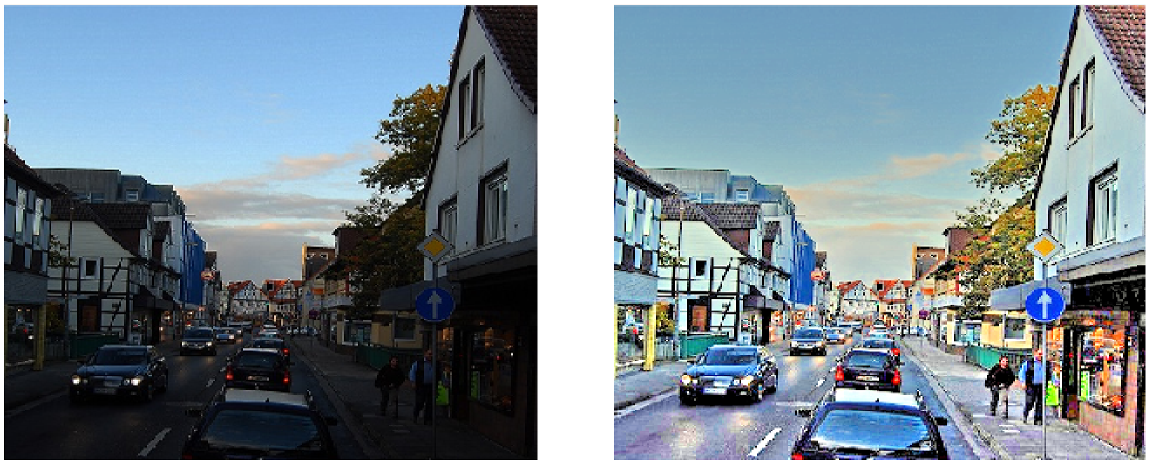
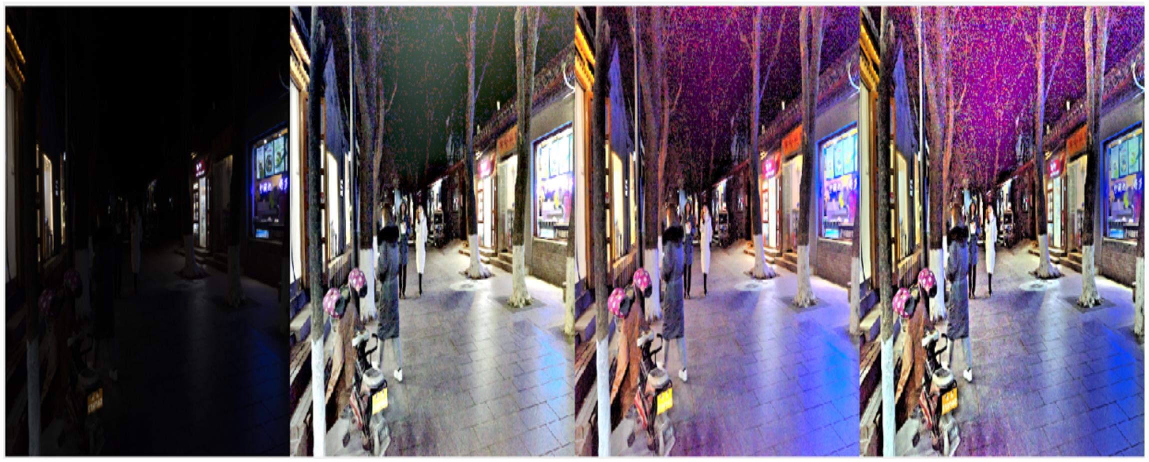


Image credit: Image from LIME dataset
Figure 2.9: Image enhancement using MSRGF

issue with MSRGF.

b) The log operator used in MSRGF create multiple issue such as it require extra check to process pixel with value 0 as log 0 is undefined. when applied onto Intensity channel the output range of log operator gives both positive as well negative values hence extending the original pixel range of intensity channel [0,1] this require extra processing to store and process these values.



(a) Original Image (b) MSRCR [8] (c) MSRCP [9] (d) MSRGF [10]

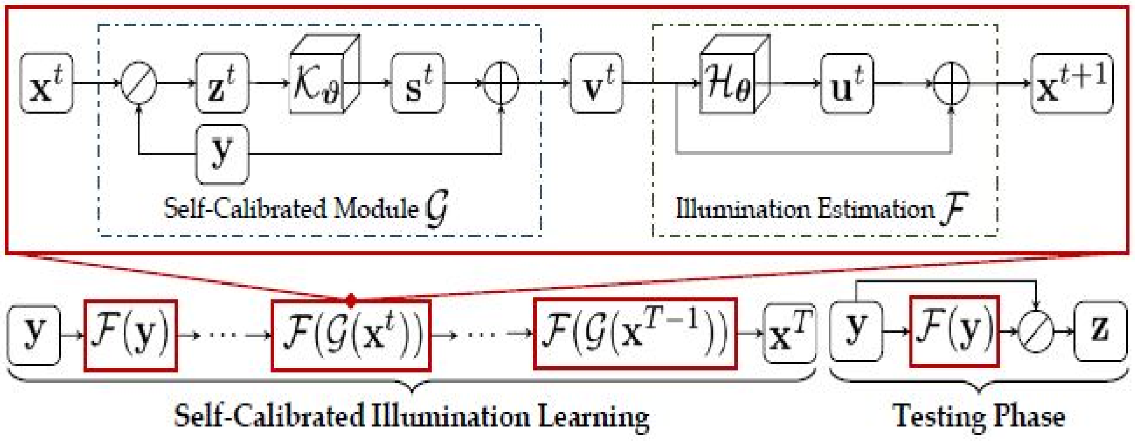
Image credit: Image taken from Dark Face Dataset
Figure 2.10: comparison between MSRCR, MSRCP and MSRGF

2.6 Self-calibrated illumination learning

Self-calibrated illumination learning framework (SCI) is a retinex theory based unsupervised method to improve the quality of image [11]. In order to learn the illumination channel. A progressive prospective model is proposed for this purpose the basic unit for the model is given as:

$$\mathcal{F}\left(\mathrm{x}^t\right) : \begin{cases} \mathbf{u}^t = \mathcal{H}_{\boldsymbol{\theta}}\left(\mathrm{x}^t\right), \mathrm{x}^0 = \mathrm{y} \\ \mathrm{x}^{t+1} = \mathbf{x}^t + \mathbf{u}^t \end{cases} \tag{2.22}$$

here y is the input image where $\mathbf{u}^t$, $\mathbf{x}^t$ represent the residual term and illumination at t-th stage $(t = 0, ..., T-1)$, and $\mathcal{H}_{\boldsymbol{\theta}}$ is the mapping function between low-light input image with its illumination component where $\boldsymbol{\theta}$ is the parameter to learn.



Self-Calibrated Module $\mathcal{G}$
Illumination Estimation $\mathcal{F}$
Self-Calibrated Illumination Learning
Testing Phase

Figure 2.11: Framework for the SCI method

In order to make the results of each stage convergent a self calibrated module is given by:

$$\mathcal{G}\left(\mathrm{x}^t\right) : \begin{cases} \mathrm{z}^t = \mathrm{y} \oslash \mathrm{x}^t \\ \mathrm{s}^t = \mathcal{K}_{\vartheta}\left(\mathrm{z}^t\right) \\ \mathbf{v}^t = \mathrm{y} + \mathbf{s}^t \end{cases} \tag{2.23}$$

here add a self-calibrated map s is added to the input image to differentiate between the input at each stage and the first stage. where $t >= 1$, converted input at each stage is $\mathbf{v}^t$ and $\mathcal{K}_{\vartheta}$ is parameterized operator with learn able parameter ϑ

$$\mathcal{F}\left(\mathrm{x}^{t}\right) \rightarrow \mathcal{F}\left(\mathcal{G}\left(\mathrm{x}^{t}\right)\right) \tag{2.24}$$

The fidelity loss $\mathcal{L}_f$ and smoothness loss $\mathcal{L}_s$ is given by :

$$\mathcal{L}_f = \sum_{t=1}^{T} \left\| \mathrm{x}^{t} - \left(\mathrm{y} + \mathrm{s}^{t-1}\right) \right\|^2 \tag{2.25}$$

$$\mathcal{L}_s = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} w_{i,j} \left| \mathbf{x}_i^t - \mathbf{x}_j^t \right| \tag{2.26}$$

Here N is total no of pixels, $\mathcal{N}(i)$ is the adjacent pixels of pixel i in a window size 5×5 and $w_{i,j}$ is the weigh given by:

$$w_{i,j} = \exp\left(- \frac{\sum_c \left(\left(\mathbf{y}_{i,c} + \mathbf{s}_{i,c}^{t-1} \right) - \left(\mathbf{y}_{j,c} + \mathbf{s}_{j,c}^{t-1} \right) \right)^2}{2\sigma^2} \right) \tag{2.27}$$

In low-light picture enhancement, dark face identification, and nighttime semantic segmentation, SCI is successful and superior. SCI is capable of handling noise suppression.

2.6.1 Issues with SCI image enhancement technique

The issues associated with MSRGF are discussed below:

a) In certain cases the color and the contrast of the enhanced image are not too good.

b) lower detail boosting as compared to other retinex based techniques.

c) Neural network based methods are generally slower because of large convolutions involved to process the image.

CHAPTER 3

Proposed Method

Continuing the work proposed by Tang *et al.* [10] where the MSRGF suffered from noise enhancement in extremely low light areas. We propose a method that replaces a log function used in MSRGF with a customized sigmoid function to mitigate the issue of noise amplification.

3.1 Sigmoid function

A detailed study of the sigmoid is carried in this section. The usefulness of sigmoid function can be found in various field such as artificial neural networks, audio signal, bio-chemistry and many others. In artificial neural the sigmoid function is used a neural activation function. In bio-chemistry sigmoid function represent closely the titration of strong acid and base. The general form of a sigmoid function is given by:

$$Sig(x) = \frac{1}{a + \exp(-bx + c)} \tag{3.1}$$

The response of the sigmoid is controlled by a, b & c where the maximum value of $Sig(x)$ is controlled by the parameter a, the slope of the function is controlled by the parameter b and parameter c controls the lateral shifts in function which

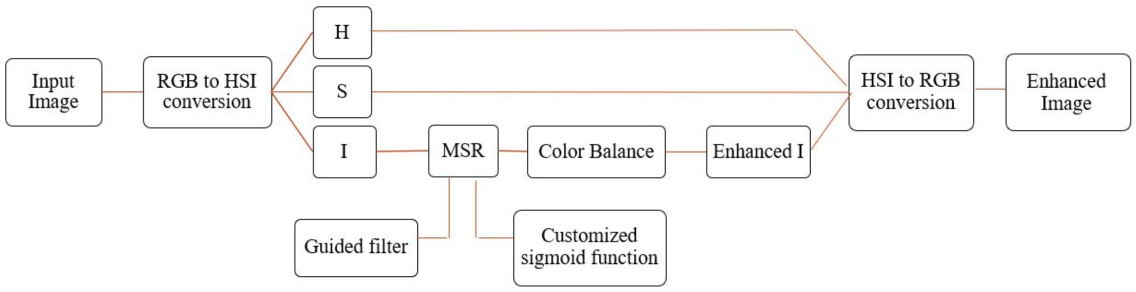


Figure 3.1: Process flow for the proposed image enhancement technique

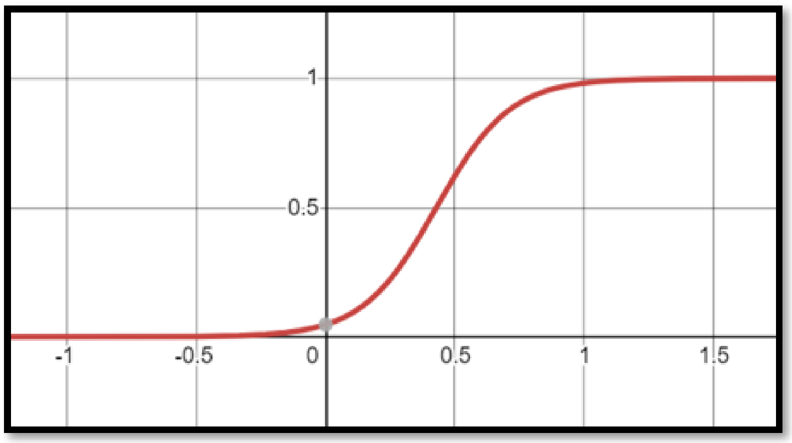


Figure 3.2: sigmoid function with a=1, b=7 and c=3

can be used to move the function away from or towards origin.

3.2 Proposed method's algorithm

The following steps have been followed to implement the proposed method:

Step 1) Convert the RGB color image into HSI Image using equation 1.2

Step 2) Compute L_1 and L_2 from equation 2.15

Step 3) Replace log with $Sig()$ function

Step 4) extract high frequency information A_1 and A_2 using equation 2.16 and 2.17

Step 5) extract low frequency information using equation 2.18

Step 6) combine the low frequency and the high frequency information using equation 2.19

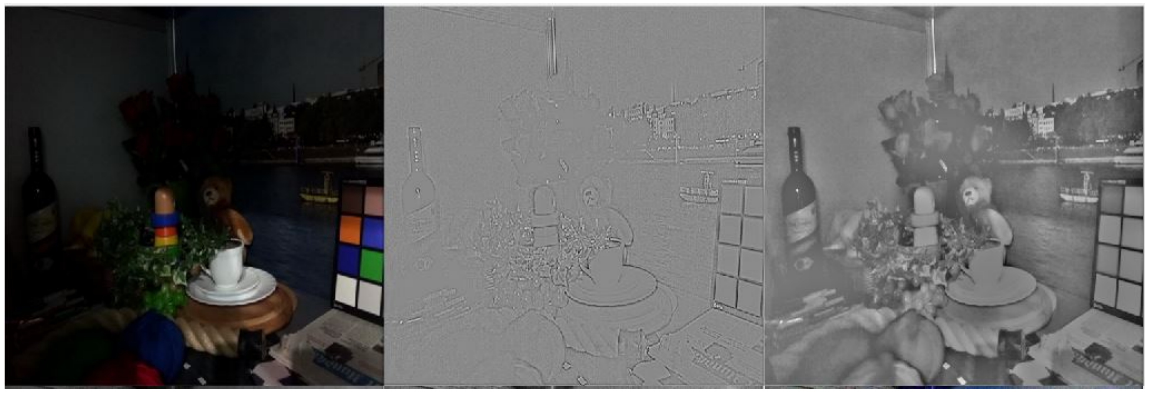
Step 7) Stretch the obtained intensity channel using equation 2.5, replace 255 with 1

Step 8) Apply a color balance technique [16] on result obtained form previous step to obtained the enhanced intensity channel

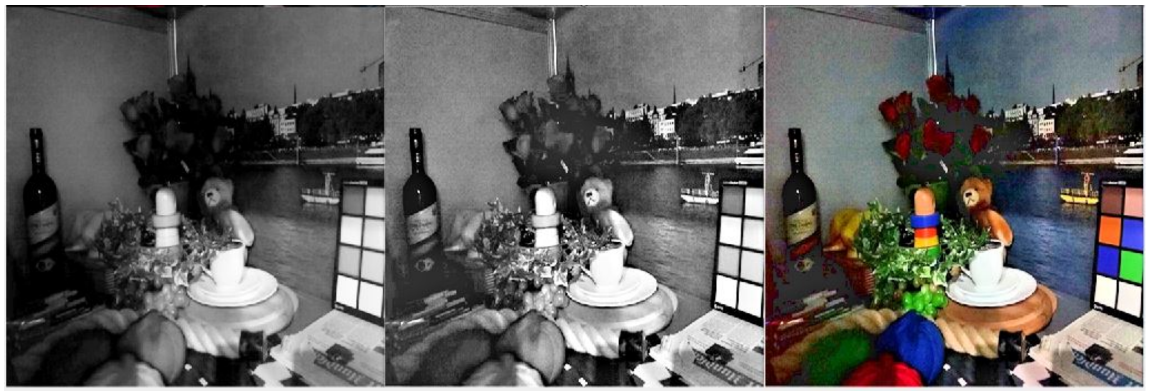
Step 9) combine the Hue (H), Saturation (S) and enhanced intensity channel

Step 10) convert image from HSI to RGB color space.

Various results obtained after applying the proposed method onto a particular image is shown in figure 3.3. The results at different obtained are first transformed to image domain for visualization purpose.



(a) Original Image (b) A_1 image (c) A_2 image



(d) LL Image (e) Enhanced intensity channel (f) Final image

Image credit: Image from LIME dataset
Figure 3.3: Results obtained at different steps of the proposed method

CHAPTER 4
FPGA Implementation

Work is in progress to implement the proposed method onto a FPGA. The FPGA implementation will allow us to understand the feasibility of the proposed method to be used in various real time applications such as traffic monitoring, video quality improvements and others. For the underlying architecture of the proposed method we will be able to explore the hardware requirements for the FPGA such as memory, clock frequency, power consumption and area. The ground work for the FPGA Implementation that has been carried till now is shown in Figure 4.1. A detailed description of each block in the FPGA architecture is given in subsequent sections.

4.1 Why FPGA implementation?

Before we jump on to conclusion regarding the FPGA implementation. It is really important to understand why FPGA is required. An FPGA is a chip that consists of a sequence of logic blocks that the user may modify and tune. As a result, these chips provide the user with far greater freedom and customization while doing specialized jobs that require quick results. FPGAs are good for parallel

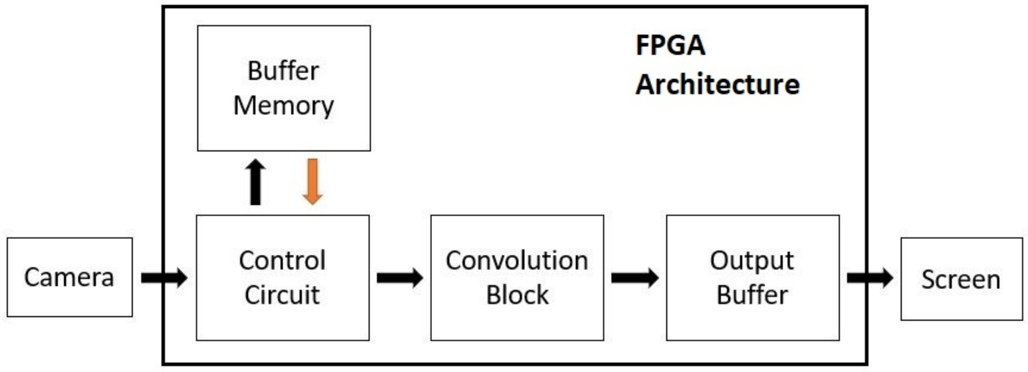


Figure 4.1: Base work of the proposed method

systems where numerous activities must be completed concurrently. FPGAs are electrically linked in the form of discrete programmable logic blocks that may be adjusted to meet the demands of the user. FPGAs are programmable hence they can have their functionality altered several times. When compared with general purpose computers, the FPGA provide several benefits these are.

a) **Flexibility** - General-purpose computers contain a predefined set of instructions that the programmer must follow. One of the most important characteristics and benefits of FPGAs is that the whole internal hardware can be reprogrammed and altered, allowing the user to choose the logic of each system block. That is, they are significantly more adaptable in their programming and may be tailored to the demands of the programmer.

b) **Execution speed** - Nothing beats a dedicated piece of hardware intended to do a particular task. As a result, a well-designed FPGA will always be faster to execute than software code running on a general-purpose CPU chip.

c) **Time Critical Processing** - General-purpose computers are typically constrained in this sense, making them unsuitable for time-critical operations, especially if the needs cannot be handled within its capabilities. FPGAs are capable of executing complicated and time-sensitive processing while also doing other vital processing jobs in parallel.

4.2 Camera

Camera is the source image/video input for the implemented FPGA design. The input image resolution for the FPGA design has been chosen to be 512×512.

4.3 Control Circuit

The input form the camera is provided to this block of the hardware design. The control circuit do the following jobs:

a) Receive the valid input pixel data stream.

b) Populate the Buffer memory with the input data.

c) Retrieve valid data back from Buffer memory and send it to the convolution block for processing.

d) Keep track of the position of current pixel under processing.

The schematic diagram of the Control circuit is shown in Figure 4.2. Various input and output signal for the control signal block are described as follows:

i) i_clk - this signal is the clock from an onboard crystal oscillator to keep all the hardware in synchronous mode.

ii) i_pixel_data_valid - this signal is an external signal provided by the camera, if this signal is high i.e. 1 then it indicates that the current pixel getting streamed by the camera is valid for use otherwise ignore the current pixel

iii) i_pixel_data[7:0] - it is an 8-bit data, it represents a single pixel value generated by the camera.

iv) i_reset - this signal provides a reset mechanism for the hardware design when high the entire circuit will get reset to idle state.

v) o_pixel_data_valid - this signal represents the validity of the output pixel, this signal will be used by the convolution block for the pixel validation.

vi) o_pixel_data[71:0] - this signal is the combined data of all the pixel that will be used for convolution with a image filter of size 3×3.

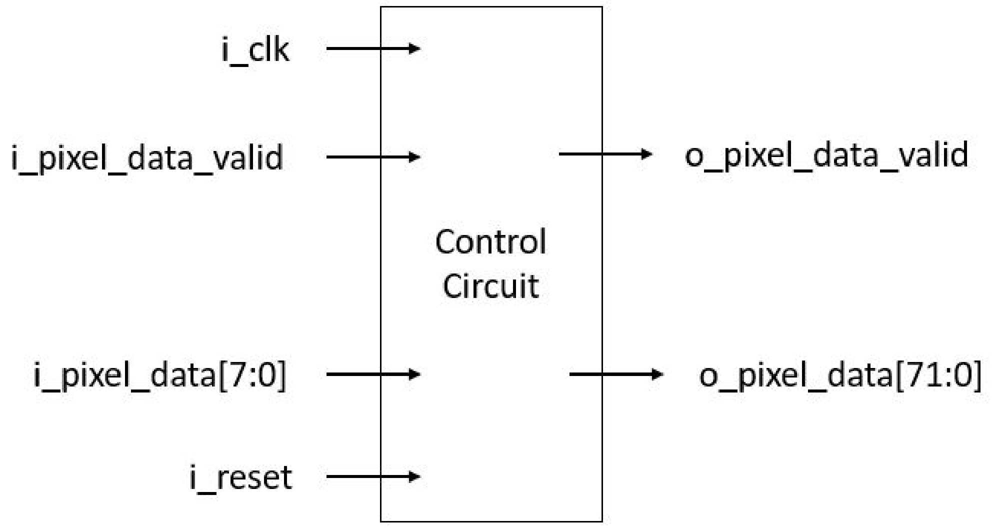


Figure 4.2: Control circuit block

4.4 Buffer Memory

The buffer memory is used to temporary store a portion of the image on the hardware. This save memory as the entire is not needed to be stored onboard the hardware. Faster convolution can be carried with such design as we can extract the required image data for each convolution from the buffer memory. Data of a row in the image is stored in a buffer line. For a 3×3 image filter the minimum required no of buffer lines is 3. A higher no of buffer lines for the given filter size can increase the performance of the design and can also help us in enabling parallel processing of the data.

The schematic diagram of a single buffer line is shown in Figure 4.3. Various input and output signal for each buffer line block are described as follows:

i) i_clk - clock signal for synchronization

ii) i_pixel_data_valid - this signal is provided by the control circuit, if this signal is high then the current pixel is valid for storage.

iii) i_pixel_data[7:0] - pixel data that needs to be stored.

iv) i_reset - this signal will clear all the data of the buffer line.

v) i_rd_data - when enabled the buffer line can be used for data read.

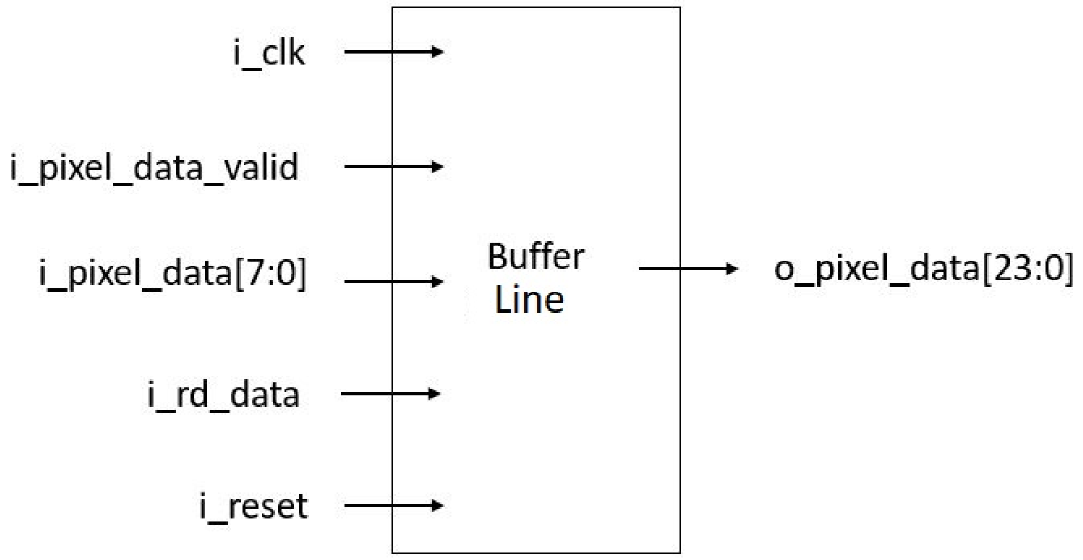


Figure 4.3: Buffer line block

vi) o_pixel_data[23:0] - this signal is the combined data of three pixel that will be used for convolution with a image filter of size 3×3.

4.5 Convolution Block

Various filter based image convolutions operations such as Gaussian blur, image sharpen, edge detection and others filters are applied onto the input image using this convolution block. The schematic diagram of the convolution block is shown in Figure 4.4. Various input and output signal for the convolution block are described as follows:

i) i_clk - clock signal for synchronization

ii) i_pixel_data_valid - this signal is provided by the control circuit, if this signal is high then the combined pixel data at the input is valid for convolution.

iii) i_pixel_data[71:0] - pixel data that is used for convolution operation.

iv) o_pixel_data_valid - this signal is used to validate the output data after convolution.

v) o_pixel_data[7:0] - result of the convolution operation.

4.6 Output Buffer

For storing the result obtained after the convolution operation we have used a output buffer. The output buffer is a FIFO (first in first out) memory block. It

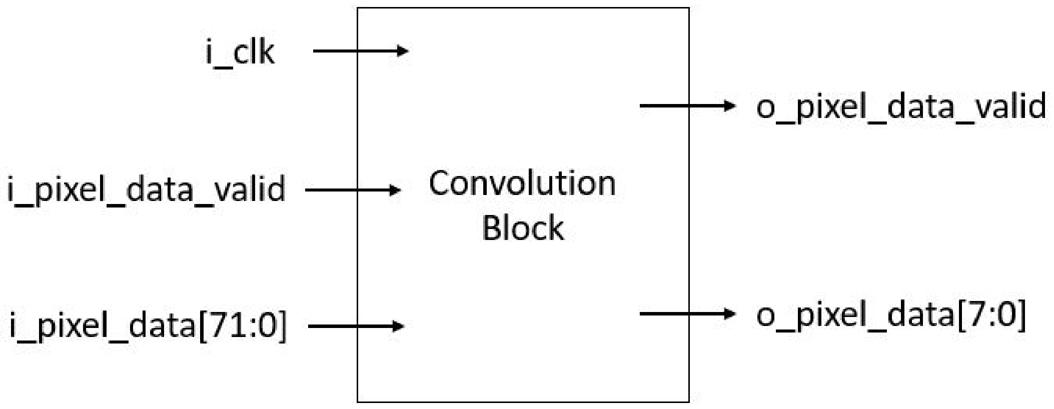


Figure 4.4: Convolution block

works on the principle os first come first serve. The data which comes first in the FIFO will be the first to get out of the FIFO.

The schematic diagram of the output buffer is shown in figure 4.5. Various input and output signal for the output buffer block are described as follows:

i) i_clk - clock signal for synchronization

ii) i_pixel_data_valid - this signal is provided by the convolution block, if this signal is high then the convolution result is valid for storage.

iii) i_pixel_data[7:0] - convolution pixel data that needs to be stored.

iv) o_pixel_data_valid - this signal is used to validate the output data after convolution.

v) o_pixel_data[7:0] - result of the convolution operation.

vi) i_reset - flush the entire FIFO memory if high

vii) i_data_ready - The FIFO will start storing the input data as soon as this signal becomes high.

viii) o_data_ready - this signal is activated when the FIFO is full and data is ready to be transferred to the next stage.

A detailed RTL schematic of the hardware implementation is shown in figure 4.6.

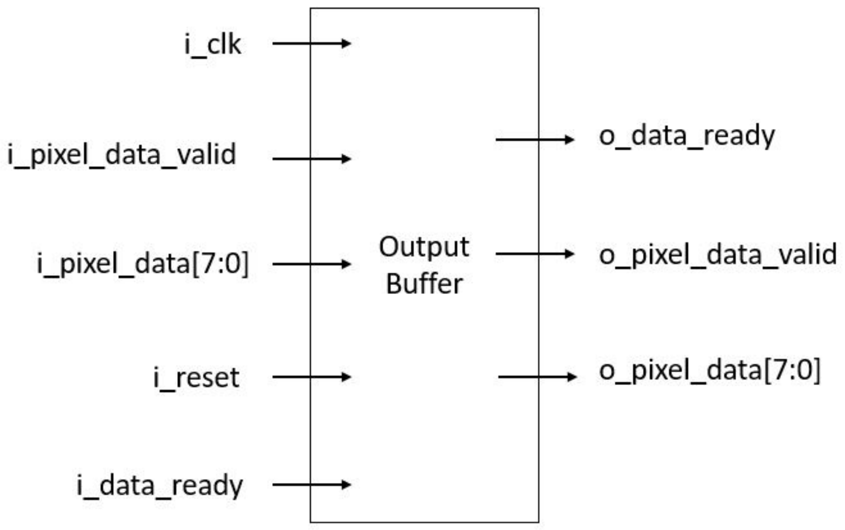


Figure 4.5: Output buffer block

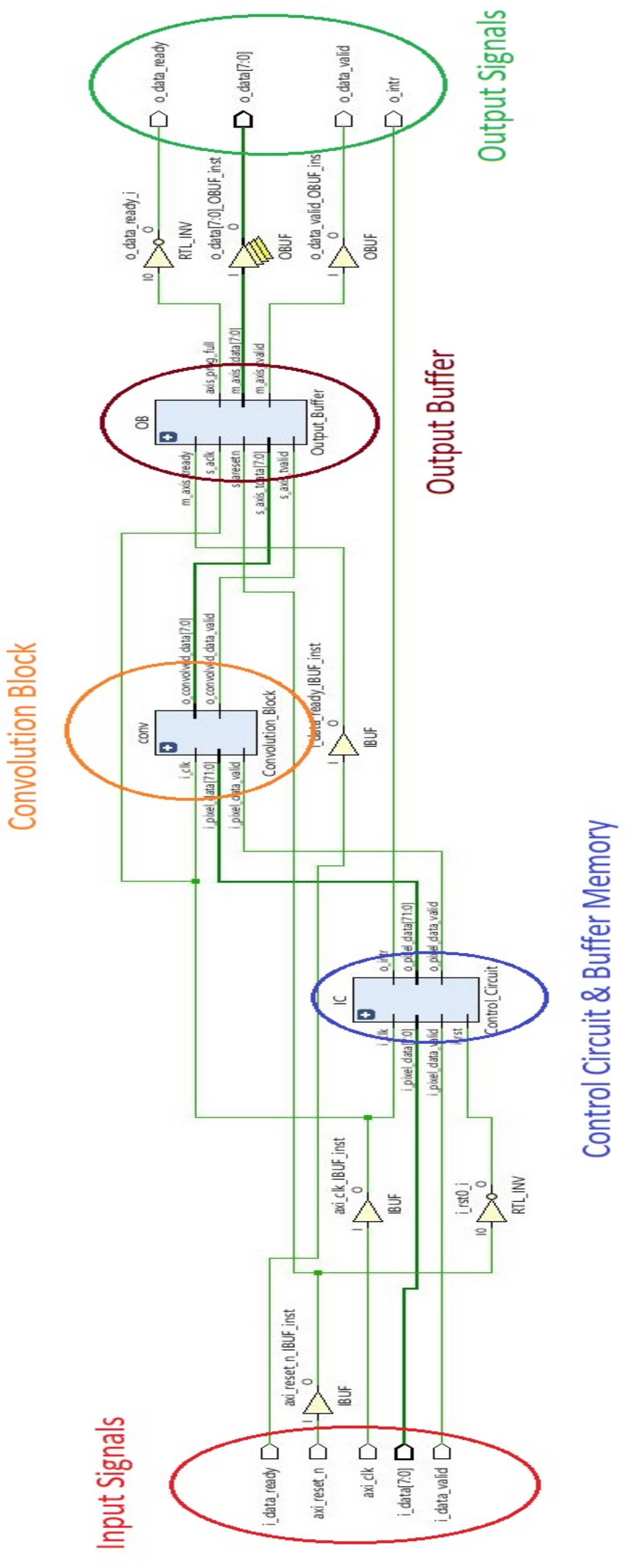


Figure 4.6: RTL design of the hardware implementation

CHAPTER 5
Results and Analysis

In this chapter we will discuss the results obtained from the proposed method. Both subjective and objective analysis of the proposed method will be performed and results will be compared with the methods previously discussed in chapter 2. Various constant values used in the methods previously discussed and the proposed method is shown in Table 5.1

5.1 Qualitative analysis of the proposed method

Qualitative analysis of any image enhancement technique is carried on the basis of how well a person or a group of persons will perceive the enhance image against the original image. Qualitative analysis includes color of the image, contrast and details in the image before and after enhancement. Based on multiple images taken from different environments. A detailed qualitative analysis is taken from figure 5.1 In order to have a lower computational time for each technique we have kept the size of the test images used to be $[512 \times 512]$. Some of these image are taken outside environment while some are taken from an in-house environment. Images taken from different environments can help us in analysing the underlying image enhancement technique for various cases.

The top left image in Figure 5.1 represents an outdoor image of a pot in low light conditions, the MSRCP method over-amplify the pixel value while in SCI method dull colors can be observed good color reproduction can be observed in MSRCR, MSRGF and the proposed method, but only in the proposed method the color environment in original is preserved.

An under water image (second image from top) in Figure 5.1 when processed with SCI method loses its color quality, while in MSRGF method over contrast can

Table 5.1: List of various constants used in image enhancement techniques

Constant used	Method			
	MSRCR [8]	MSRCP [9]	MSRGF [10]	Proposed
c_1	15	15	-	-
c_2	80	80	-	-
c_3	250	250	-	-
r_1	-	-	15	5
r_2	-	-	80	100
α	125	-	-	-
β	46	-	-	-
γ	-	-	4.5	-
s_1	-	1	-	2
s_2	-	1	-	2
a	-	-	-	1
b	-	-	-	4
c	-	-	-	-7
ϵ_k	-	-	0.04	0.04

be observed , for the MSRCP method strong colors are produced in the enhanced image, only in MSRCR and the proposed method good colors are reproduced. The results of MSRCR in this case are visually more appealing.

The third image from top in Figure 5.1 is an in-door image, artifacts are produced in the enhanced image using MSRGF method, while the MSRCP method provides a high contrast image. The SCI and MSRCR method avoids the artifacts from affecting this image but the overall color is still dull when compared with the proposed method.

Third last image in Figure 5.1 in an outdoor traffic image for this image both MSRCP and the proposed method provide good color in the enhanced image. MSRCR output is dull amongst all other methods in cases of SCI certain regions of the image contains good colors but some regions such as the sky appears very dull. the details are very well preserved in MSRGF as well as the proposed method.

The second last image is an an out-door image, for this image both MSRCR and SCI method suffers from poor colors contrast in the enhanced image, the results of the proposed method and the MSRCP method are comparatively good in terms of color and contrast, The MSRGF method is producing high details boosting which in this case is dominating the color quality of the image.

The last image is taken at very low light conditions, good colors are produced when MSRCR, MSRCP and MSRGF techniques are applied onto the original image, but noise amplification can also be observed in the image enhanced by these methods. The SCI technique provide Superior noise suppression, followed by the proposed method. both of these methods falls behind in terms of contrast and color when compared to other methods.

5.2 Quantitative analysis of the proposed method

For different individuals the standard for good color, good contrast can differ hence it become very important to analyse any image enhancement technique both qualitative and quantitative to show case is benefits over other image enhancement methods.

For quantitative analysis we have used multiple image quality measures which are as follows:

a) Entropy - Entropy is used to characterize the texture of the enhanced image. To compute entropy we use equation 5.1

b) PSNR - Peak signal to noise ratio is ratio of maximum power of the image signal with the noise power present in the image. PSNR is given by equation 5.2

c) SSIM - Structural similarity index is used to measure the similarity between the original image and the enhanced image. This measure was proposed by Wang *et al.* [17] and it primarily focuses on the detail preservation property of the image enhancement algorithm.

d) BRISQUE - Blind/Reference-less image spatial quality evaluator [18] is a distortion generic no-reference image quality assessment model that operates in the spatial domain and is based on natural scene statistics. It instead employs scene statistics of regionally normalised brightness coefficients to assess probable losses of ’naturalness’ in the image due to the presence of distortions, resulting in a holistic measure of quality.

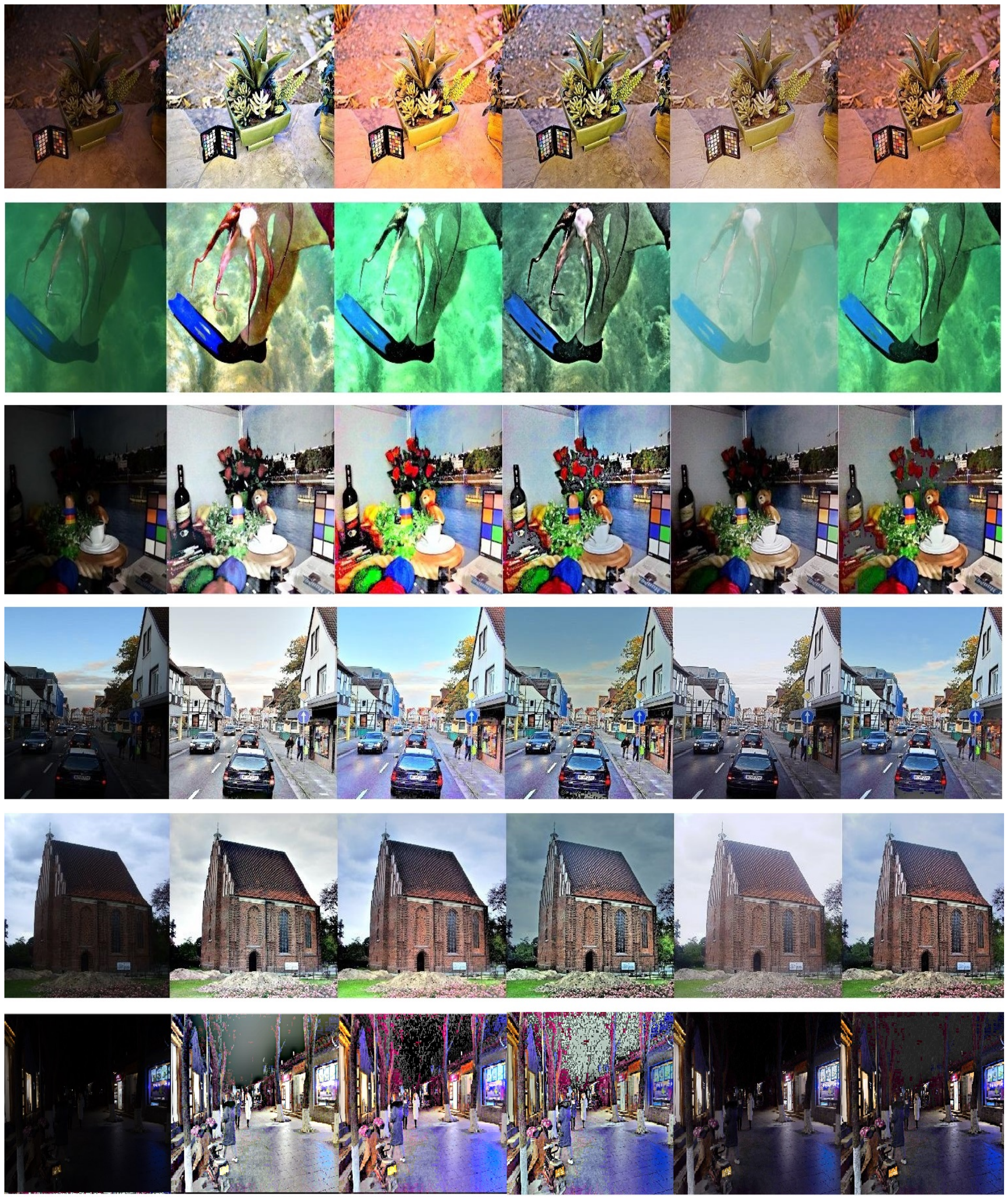


Image credit: Image 1,3,4 & 5 from LIME dataset, Image 2 by Catalina Sbert, Image 6 from Data Face dataset

Figure 5.1: Results obtained from different image processing algorithms to carry out a detailed qualitative analysis

e) PIQE - Perception based Image Quality Evaluator [19] is a blind quality assessment that measures the local variance of perceptibly distorted blocks and estimates block-wise distortion to compute the quality score.

f) NIQE - Naturalness Image Quality Evaluator [20] is also a blind image quality assessment that solely uses quantifiable deviations from statistical regularities detected in natural pictures, with no training or exposure to distorted images.

Formulas for various measures used is given below:

$$Entropy = -\sum_{i=0}^{255} p_i \log_2 p_i \tag{5.1}$$

where p_i is probability of pixel i obtained form normalized histogram of image.

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \tag{5.2}$$

here MSE is given as,

$$MSE = \frac{1}{M \times N \times C} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{C} (I_{i,j,k} - \hat{I}_{i,j,k})^2 \tag{5.3}$$

here $M \times N \times C$ represents total no of data points in a image with M rows, N columns and C color channels. $I_{i,j,k}$ is the data point in the original image and $\hat{I}_{i,j,k}$ is the data point in the enhanced image.

The SSIM index is given by:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{5.4}$$

here $C_1 = (0.01 * 255)^2$ and $C_2 = (0.03 * 255)^2$ where $\mu_x, \mu_y, \sigma_x, \sigma_y$, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. The range for each measure is given in Table 5.2. For Entorpy, PSNR, and SSIM higher score is preferred. But for measures such as BRIQSUE, NIQE and PIQE lower score represents higher quality of the enhanced image. Finally using these measures we compared our method with other image enhancement techniques. The detailed comparison is shown in table 5.3. Based on this the best and the second best score for each image quality measure has been shown for different image enhancement techniques applied to different images is shown in Table 5.4

Table 5.2: Range of various measure used for analysis of image enhancement techniques

Quality measure used	Range	
	Lowest	Highest
Entropy ↑	0	8
PSNR ↑ (in db)	0	inf
SSIM ↑	0	1
BRISQUE ↓	0	100
PIQE ↓	0	100
NIQE ↓	0	100

Note : For each measure ↑ indicates higher value is preferable and ↓ indicates lower value is preferable.

Table 5.3: Comparison between different image enhancement technique when applied on multiple images, using different image quality measures

Measure	Method	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Avg
Entropy	MSRCR [8]	6.365	6.618	7.039	6.394	6.946	7.15	6.752
	MSRCP [9]	6.761	7.173	7.102	7.059	7.709	7.123	7.154
	MSRGF [10]	7.089	7.108	7.295	7.159	7.32	7.573	7.257
	SCI [11]	7.598	6.914	7.436	7.69	7.453	6.549	7.273
	Proposed	7.527	7.591	7.334	7.593	7.521	6.579	7.358
PSNR	MSRCR [8]	5.768	8.293	5.148	6.958	9.255	5.817	6.873
	MSRCP [9]	5.559	8.171	4.96	6.756	9.284	6.417	6.857
	MSRGF [10]	9.072	12.83	6.936	8.592	9.352	6.129	8.819
	SCI [11]	8.827	8.261	11.48	11.82	11.34	15.64	11.227
	Proposed	10.56	12.68	10.83	11.35	12.47	14.37	12.042
SSIM	MSRCR [8]	0.183	0.228	0.203	0.27	0.294	0.069	0.207
	MSRCP [9]	0.267	0.483	0.169	0.439	0.508	0.083	0.324
	MSRGF [10]	0.436	0.559	0.163	0.45	0.461	0.077	0.357
	SCI [11]	0.454	0.572	0.387	0.428	0.446	0.389	0.446
	Proposed	0.436	0.682	0.281	0.497	0.54	0.201	0.489
BRISQUE	MSRCR [8]	40.76	11.99	35.29	32.01	35.98	27.79	30.635
	MSRCP [9]	41.28	11.33	37.33	32.92	35.16	37.47	32.582
	MSRGF [10]	40.31	41.82	36.67	35.55	31.13	25.19	35.109
	SCI [11]	42.56	15.69	49.17	31.97	32.84	15.98	31.366
	Proposed	38.93	18.42	34.02	28.3	37.62	32.28	31.595
NIQE	MSRCR [8]	3.832	4.088	3.159	3.904	4.307	2.86	3.691
	MSRCP [9]	3.714	3.911	3.427	4.045	4.784	2.992	3.812
	MSRGF [10]	4.157	4.724	3.825	3.887	4.371	2.98	3.990
	SCI [11]	5.597	4.542	4.727	4.576	4.762	2.719	4.487
	Proposed	3.999	3.705	3.769	3.41	3.925	3.043	3.641
PIQE	MSRCR [8]	45.89	27.84	54.85	56.61	53.37	47.03	47.596
	MSRCP [9]	42.64	23.99	55.33	55.15	54.02	46.05	46.197
	MSRGF [10]	41.26	33.32	48.55	54.57	53.32	46.65	46.275
	SCI [11]	61.18	5.392	72.62	62.16	47.18	44.91	48.907
	Proposed	43.58	17.15	48.08	51.89	48.3	43.8	42.133

Table 5.4: Best and second best score obtained form image enhancement technique for each measure when applied to different images

Image	Score	Measure					
		Entropy	PSNR	SSIM	BRISQUE	NIQE	PIQE
1	1st	SCI [11]	Proposed	SCI [11]	Proposed	MSRCP [9]	MSRCR [8]
	2nd	Proposed	MSRGF [10]	Proposed	MSRGF [10]	MSRCR [8]	MSRCP [9]
2	1st	Proposed	MSRGF [10]	Proposed	MSRCP [9]	Proposed	SCI [11]
	2nd	MSRCP [9]	Proposed	SCI [11]	MSRCR [8]	MSRCP [9]	Proposed
3	1st	SCI [11]	SCI [11]	Proposed	Proposed	MSRCR [8]	Proposed
	2nd	Proposed	Proposed	SCI [11]	MSRCR [8]	MSRCP [9]	MSRGF [10]
4	1st	SCI [11]	SCI [11]	Proposed	Proposed	Proposed	Proposed
	2nd	Proposed	Proposed	MSRGF [10]	SCI [11]	MSRCR [8]	MSRGF [10]
5	1st	MSRCP [9]	Proposed	Proposed	MSRGF [10]	Proposed	Proposed
	2nd	Proposed	SCI [11]	MSRCP [9]	SCI [11]	MSRCR [8]	SCI [11]
6	1st	MSRGF [10]	SCI [11]	SCI [11]	SCI [11]	SCI [11]	Proposed
	2nd	MSRCR [8]	Proposed	Proposed	MSRGF [10]	MSRCR [8]	SCI [11]

For most of the images the proposed method gives Superior average results in terms of Entropy, PIQE, NIQE, PSNR and SSIM when compared with other methods. The average Entropy improvement is 1.1% form the second best method. For PSNR 7.25% average improvements are observed when compared with second best method. Similarly in terms of SSIM the average improvements is 9.6%, for NIQE it is 1.13% and for PIQE the improvement is 8.8%. For BRISQUE the proposed method falls behind to other methods with 3.13% lower average score when compared with method performing best for this measure.

CHAPTER 6

FPGA Simulations and Results

The FPGA architecture for the image processing algorithm has been designed in verilog on Xilinix Vivado v2018.3. The verliog code for each module in FPGA implementation is provided in Appendix Chapter B. For simulating the FPGA architecture the image in figure 6.4 has been used. The detailed RTL schematic of the hardware implementation is shown in figure 4.6 The FPGA Design has following design constraints :

a) Total number of Line Buffer Used - 4

b) Depth of each Line Buffer - 512

c) Input image dimensions 512 $\times$ 512

d) Image filter dimensions 3 $\times$ 3

e) Clock period - 10ns

f) Output Buffer depth - 16

g) FPGA used - Nexys 4-DDR xc7a100tcsg324-1

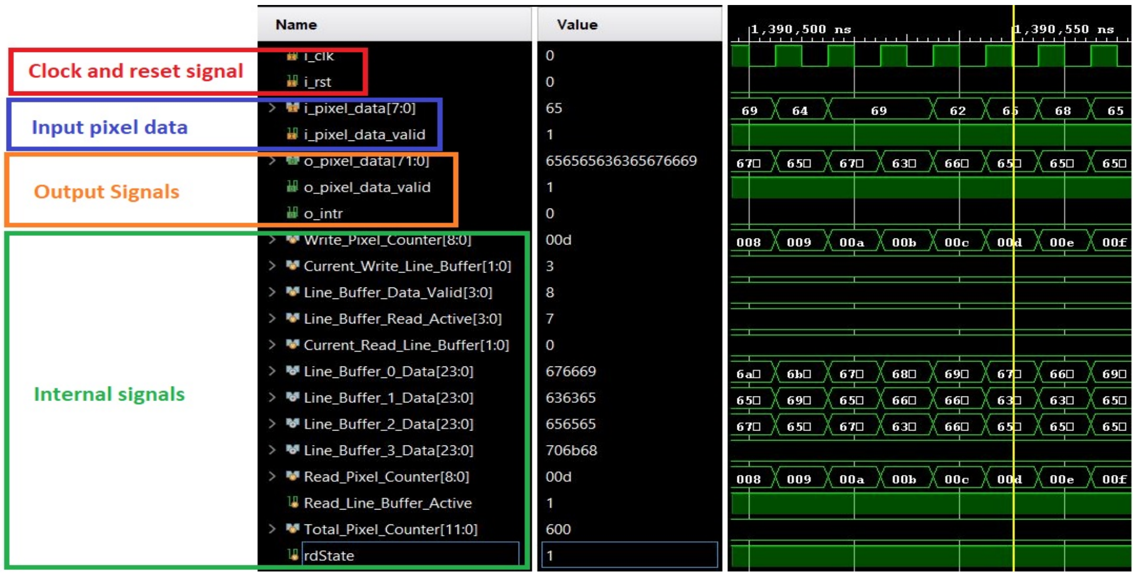
Table 6.1: Image filter 1

1	0	-1
2	0	-2
1	0	-1

Table 6.2: Image filter 2

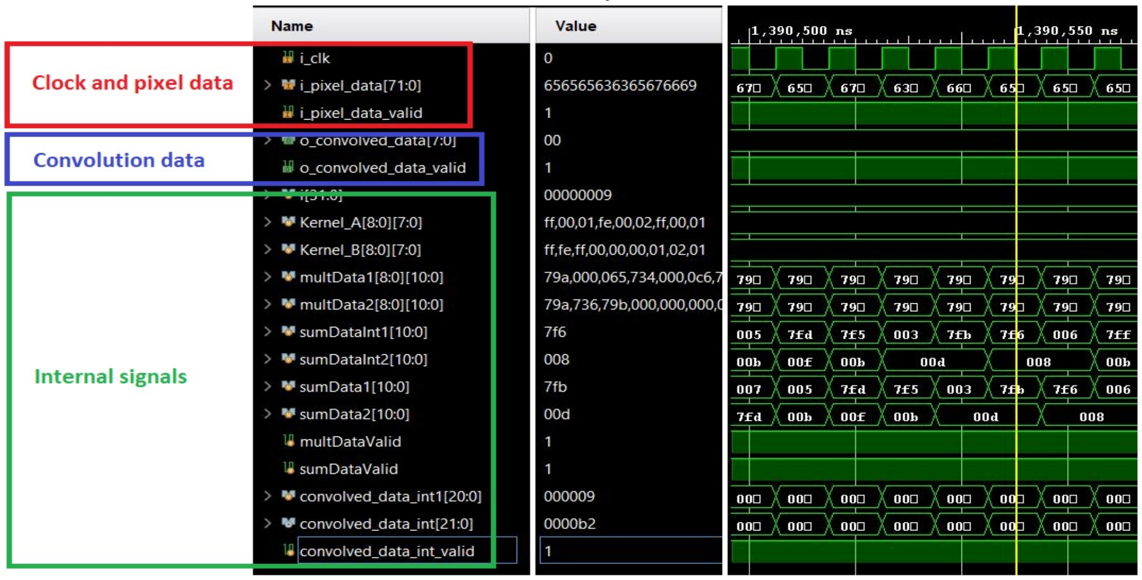
1	2	1
0	0	0
-1	-2	-1

The results obtained from the simulations are shown in figure 6.1



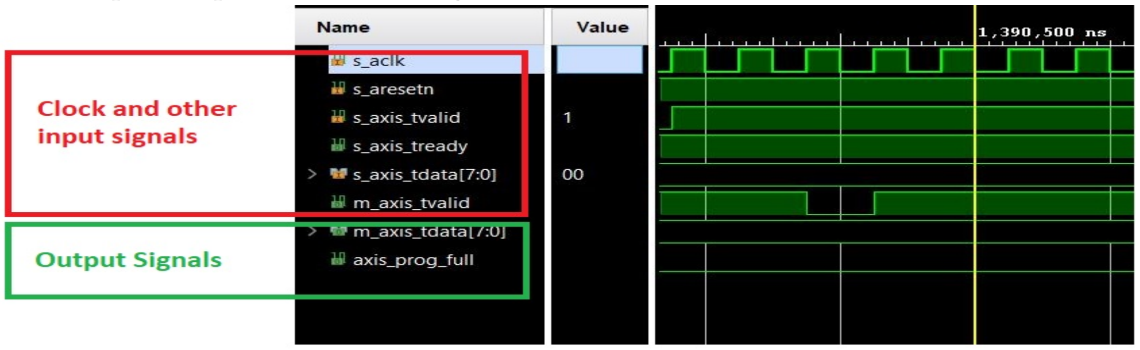
Name	Value
i_clk	0
i_rst	0
i_pixel_data[7:0]	65
i_pixel_data_valid	1
o_pixel_data[71:0]	656565636365676669
o_pixel_data_valid	1
o_intr	0
Write_Pixel_Counter[8:0]	00d
Current_Write_Line_Buffer[1:0]	3
Line_Buffer_Data_Valid[3:0]	8
Line_Buffer_Read_Active[3:0]	7
Current_Read_Line_Buffer[1:0]	0
Line_Buffer_0_Data[23:0]	676669
Line_Buffer_1_Data[23:0]	636365
Line_Buffer_2_Data[23:0]	656565
Line_Buffer_3_Data[23:0]	706b68
Read_Pixel_Counter[8:0]	00d
Read_Line_Buffer_Active	1
Total_Pixel_Counter[11:0]	600
rdState	1

(a) Input, output and internal signals simulation waveform for control circuit and buffer memory



Name	Value
i_clk	0
i_pixel_data[71:0]	656565636365676669
i_pixel_data_valid	1
o_convolved_data[7:0]	00
o_convolved_data_valid	1
i[31:0]	00000009
Kernel_A[8:0][7:0]	ff,00,01,fe,00,02,ff,00,01
Kernel_B[8:0][7:0]	ff,fe,ff,00,00,00,01,02,01
multData1[8:0][10:0]	79a,000,065,734,000,0c6,7
multData2[8:0][10:0]	79a,736,79b,000,000,000,0
sumDataInt1[10:0]	7f6
sumDataInt2[10:0]	008
sumData1[10:0]	7fb
sumData2[10:0]	00d
multDataValid	1
sumDataValid	1
convolved_data_int1[20:0]	000009
convolved_data_int[21:0]	0000b2
convolved_data_int_valid	1

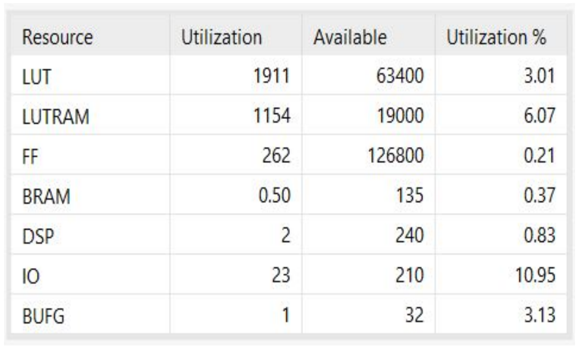
(b) Input, output and internal signals simulation waveform for convolution block



Name	Value
s_aclk	
s_aresetn	
s_axis_tvalid	1
s_axis_tready	
s_axis_tdata[7:0]	00
m_axis_tvalid	
m_axis_tdata[7:0]	
axis_prog_full	

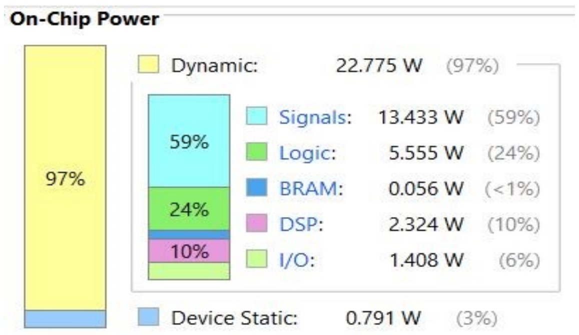
(c) Input and output signals simulation waveform for output buffer memory

Figure 6.1: Simulation waveforms for various design block in the FPGA design



Resource	Utilization	Available	Utilization %
LUT	1911	63400	3.01
LUTRAM	1154	19000	6.07
FF	262	126800	0.21
BRAM	0.50	135	0.37
DSP	2	240	0.83
IO	23	210	10.95
BUFG	1	32	3.13

Figure 6.2: Resource utilization of the FPGA implementation

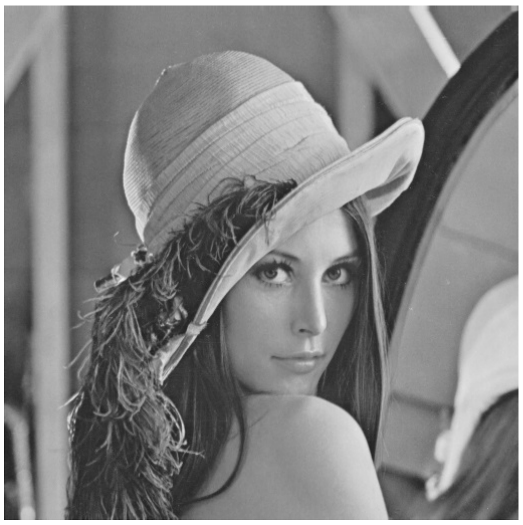


On-Chip Power

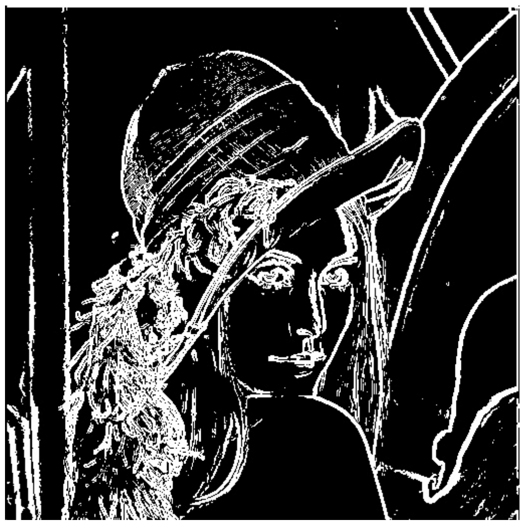
🟨	Dynamic:	22.775 W	(97%)
🟦	Signals:	13.433 W	(59%)
🟩	Logic:	5.555 W	(24%)
🟦	BRAM:	0.056 W	(<1%)
🟪	DSP:	2.324 W	(10%)
🟩	I/O:	1.408 W	(6%)
🟦	Device Static:	0.791 W	(3%)

Figure 6.3: Power consumption of the FPGA implementation

The overall resource requirements for the design is shown in figure 6.2. For the hardware implementation a total of 24.57% of all the hardware resource available on the nexys 4-DDR is consumed. The overall power requirements for the design is shown in figure 6.3.



(a) Original lenna image



(b) Filtered lenna image

Figure 6.4: Simulations results

The amount of time taken to process the input image with the underlying

FPGA implementation is around 2.6ms. With the same design constraints used the FPGA implementation can process 380 frames per seconds.

CHAPTER 7
Conclusions

The noise enhancement issue associated with various retinex based image enhancement algorithms can be tackled by using the proposed method, the sigmoid function used in the proposed method provide better dynamic range compression as compared to log operator. After rigorous trials have come up with general values for the constants a, b and c used in sigmoid function for near all types of images. The proposed performs better in terms of visual image quality.

Both qualitative and quantitative analysis of the proposed method has been carried where the proposed method provide better color and contrast results than the other image enhancement techniques we used for the comparison. In terms of qualitative analysis the proposed method when compared with other methods provides relatively better scores for measures such as Entropy, PSNR, SSIM, NIQE and PIQE. In terms of BRISQUE the proposed method has the second best quality score.

When used for particular type of images such as under water imaging, medical imaging, traffic signal imaging and others the values associated with sigmoid function can be explored for the specific purpose.

We have also implemented a basic hardware design for FPGA based image processing .The design is capable of processing 380 frames per second of input image size 512×512, the hardware design only uses a fourth of the total hardware resources available on the Nexys 4-DDR while consuming 22.775 watt of power.

References

[1] H. Yeganeh and Z. Wang, "Objective assessment of tone mapping algorithms," in *2010 IEEE International Conference on Image Processing*, 2010, pp. 2477–2480.

[2] G. Srivastava and T. K. Rawat, "Histogram equalization: A comparative analysis amp; a segmented approach to process digital images," in *2013 Sixth International Conference on Contemporary Computing (IC3)*, 2013, pp. 81–85.

[3] X. Guan, S. Jian, P. Hongda, Z. Zhiguo, and G. Haibin, "An image enhancement method based on gamma correction," in *2009 Second International Symposium on Computational Intelligence and Design*, vol. 1, 2009, pp. 60–63.

[4] J. Stark, "Adaptive image contrast enhancement using generalizations of histogram equalization," *IEEE Transactions on Image Processing*, vol. 9, no. 5, pp. 889–896, 2000.

[5] S.-C. Huang, F.-C. Cheng, and Y.-S. Chiu, "Efficient contrast enhancement using adaptive gamma correction with weighting distribution," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 1032–1041, 2013.

[6] E. H. Land and J. J. McCann, "Lightness and retinex theory," *J. Opt. Soc. Am.*, vol. 61, no. 1, pp. 1–11, Jan 1971.

[7] Z. Rahman, D. Jobson, and G. Woodell, "Multi-scale retinex for color image enhancement," in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 3, 1996, pp. 1003–1006 vol.3.

[8] Z. ur Rahman, D. J. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 100 – 110, 2004.

[9] A. B. Petro, C. Sbert, and J.-M. Morel, "Multiscale retinex," *Image Processing On Line*, pp. 71–88, 2014.

[10] J. M. Z. Z. Shi Tang, Mingjie Dong and C. Li, “Color image enhancement based on retinex theory with guided filter,” in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 5676–5680.

[11] L. Ma, T. Ma, R. Liu, X. Fan, and Z. Luo, “Toward fast, flexible, and robust low-light image enhancement,” 2022.

[12] X. Luo, H.-Q. Zeng, Y. Wan, X.-B. Zhang, Y.-P. Du, and T. M. Peters, “Endoscopic vision augmentation using multiscale bilateral-weighted retinex for robotic surgery,” *IEEE Transactions on Medical Imaging*, vol. 38, no. 12, pp. 2863–2874, 2019.

[13] H. Huang, Y. Jin, and G. Li, “An improved retinex algorithm for underwater image enhancement based on hsv model,” in *2021 International Conference on Sensing, Measurement Data Analytics in the era of Artificial Intelligence (ICSMD)*, 2021, pp. 1–5.

[14] H. Wen, F. Dai, and D. Wang, “A survey of image dehazing algorithm based on retinex theory,” in *2020 5th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, 2020, pp. 38–41.

[15] P. Jidesh and I. P. Febin, “A perceptually inspired variational model for enhancing and restoring remote sensing images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 2, pp. 251–255, 2021.

[16] N. Limare, J.-L. Lisani, J.-M. Morel, A. B. Petro, and C. Sbert, “Simplest Color Balance,” *Image Processing On Line*, vol. 1, pp. 297–315, 2011.

[17] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[18] A. Mittal, A. K. Moorthy, and A. C. Bovik, “No-reference image quality assessment in the spatial domain,” *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[19] V. N, P. D, M. C. Bh, S. S. Channappayya, and S. S. Medasani, “Blind image quality evaluation using perception based features,” in *2015 Twenty First National Conference on Communications (NCC)*, 2015, pp. 1–6.

[20] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a “completely blind” image quality analyzer,” *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2013.

CHAPTER A
MATLAB codes

A.1 Proposed method

```
function [enhanced_image] = Proposed_Method(Img,a,b,c)
    % Img - > input RGB color Image
    % a,b,c -> parameters for the sigmoid function
    my_img = rgb2hsv(Img);  % RGB to HSI conversion
    Sum = my_img(:,:,3);     % extract intensity channel
    Z1 = sigmoidfun(Sum,a,b,c);
    L1 = imguidedfilter(Sum,'NeighborhoodSize',...
            [5,5],'DegreeOfSmoothing',0.04);
    L2 = imguidedfilter(Sum,'NeighborhoodSize',...
            [100,100],'DegreeOfSmoothing',0.04);
    A1 = Z1 - sigmoidfun(L1,a,b,c);
    A2 = Z1 - sigmoidfun(L2,a,b,c);
    A3 = Z1 - sigmoidfun(A1,a,b,c);
    ttemp = (1.2*A1+1.2*A2+A3);
    ttemp = Img_Domain(ttemp);
    ttempp = single(ColorBalance(ttemp,1,1))/255;
    my_img(:,:,3) = ttempp;
    enhanced_image = hsv2rgb(my_img);
end
```

A.2 Sigmoid function

```
function [outexp] = sigmoidfun(Img,a,b,c)
    [dim1,dim2] = size(Img);
    outexp = Img;
    for i = 1:dim1
```

```
        for j = 1:dim2
                outexp(i,j) = 1./(a + exp(-b*Img(i,j)+c));
        end
    end
end
```

A.3 Linear stretching

```
function [Out_Img] = Img_Domain(Img)
    Img = single(Img);
    Out_Img = zeros(size(Img));
    maxel = max(Img,[],[1,2]);
    minel = min(Img,[],[1,2]);
    for i = 1:size(Img,3)
        Out_Img(:,:,i) = 255.*(Img(:,:,i)-minel(i)) ...
            ./(maxel(i)-minel(i));
    end
    Out_Img = uint8(Out_Img);
end
```

A.4 Color balance

```
 function [outimg] = ColorBalance(Img,s1,s2)
    outimg = zeros(size(Img));
    cumhist_y = zeros(256,size(Img,3));
    for i=1:size(Img,3)
        crtt = imhist(Img(:,:,i),256);
        sum_t = 0;
        for p = 1:256
            sum_t = sum_t + crtt(p);
            cumhist_y(p,i) = sum_t;
        end
        cumhist_y(:,i) = cumhist_y(:,i)/sum_t;
    end
    for i=1:size(Img,3)
         g = find( cumhist_y(:,i) > s1/100,1,'first' );
         gg = find( cumhist_y(:,i) < 1-s2/100,1, 'last' );
         temp = single(Img(:,:,i));
         temp(temp<g) = g;
         temp(temp>gg) = gg;
         outimg(:,:,i) = 255.*(temp-g)./(gg-g);
    end
    outimg = uint8(outimg);
 end
```

CHAPTER B
Verilog Codes

B.1 Control Circuit

```
`timescale 1ns / 1ps
module Control_Circuit(
input                       i_clk,
input                       i_rst,
input [7:0]                 i_pixel_data,
input                       i_pixel_data_valid,
output reg [71:0]           o_pixel_data,
output                      o_pixel_data_valid,
output reg                  o_intr
);

reg [8:0] Pixel_Counter
reg [1:0] Current_Write_LineBuffer;
reg [3:0] LineBuffer_Data_Valid;
reg [3:0] LineBuffer_Read_Data;
reg [1:0] Current_Read_LineBuffer;
wire [23:0] LineBuffer_0_Data;
wire [23:0] LineBuffer_1_Data;
wire [23:0] LineBuffer_2_Data;
wire [23:0] LineBuffer_3_Data;
reg [8:0] Read_Counter;
reg Read_Line_Buffer;
reg [11:0] Total_Pixel_Counter;
reg Machine_State;
```

```
localparam IDLE = 'b0,
           RD_BUFFER = 'b1;

assign o_pixel_data_valid = Read_Line_Buffer;

always @(posedge i_clk) begin
    if(i_rst)
        Total_Pixel_Counter <= 0;
    else    begin
        if(i_pixel_data_valid & !Read_Line_Buffer)
            Total_Pixel_Counter <= Total_Pixel_Counter + 1;
        else if(!i_pixel_data_valid & Read_Line_Buffer)
            Total_Pixel_Counter <= Total_Pixel_Counter - 1;
    end
end

always @(posedge i_clk) begin
    if(i_rst)     begin
        Machine_State <= IDLE;
        Read_Line_Buffer <= 1'b0;
        o_intr <= 1'b0;
    end     else     begin
        case(Machine_State)
            IDLE:begin
                o_intr <= 1'b0;
                if(Total_Pixel_Counter >= 1536)     begin
                    Read_Line_Buffer <= 1'b1;
                    Machine_State <= RD_BUFFER;
                end
            end
            RD_BUFFER:begin
                if(Read_Counter == 511)    begin
                    Machine_State <= IDLE;
                    Read_Line_Buffer <= 1'b0;
                    o_intr <= 1'b1;
                end
            end
        endcase
    end
end

always @(posedge i_clk) begin
    if(i_rst)
        pixelCounter <= 0;
    else     begin
        if(i_pixel_data_valid)
            pixelCounter <= pixelCounter + 1;
    end
end


always @(posedge i_clk)
begin
    if(i_rst)
        Current_Write_LineBuffer <= 0;
    else     begin
        if(pixelCounter == 511 & i_pixel_data_valid)
            Current_Write_LineBuffer <= Current_Write_LineBuffer+1;
    end
end


always @(*) begin
    LineBuffer_Data_Valid = 4'h0;
    LineBuffer_Data_Valid[Current_Write_LineBuffer] =
                i_pixel_data_valid;
end

always @(posedge i_clk) begin
    if(i_rst)
        Read_Counter <= 0;
    else  if(Read_Line_Buffer)
         Read_Counter <= Read_Counter + 1;
end
```

```
always @(posedge i_clk) begin
    if(i_rst)      begin
        Current_Read_LineBuffer<= 0;
    end      else      begin
        if(Read_Counter == 511 & Read_Line_Buffer)
            Current_Read_LineBuffer<= Current_Read_LineBuffer+ 1;
    end
end


always @(*) begin
    case(currentRdLineBuffer)
        0:begin
            o_pixel_data = {LineBuffer_2_Data,LineBuffer_1_Data,
                                    LineBuffer_0_Data};
        end
        1:begin
            o_pixel_data = {LineBuffer_3_Data,LineBuffer_2_Data,
                                    LineBuffer_1_Data};
        end
        2:begin
            o_pixel_data = {LineBuffer_0_Data,LineBuffer_3_Data,
                                    LineBuffer_2_Data};
        end
        3:begin
            o_pixel_data = {LineBuffer_1_Data,LineBuffer_0_Data,
                                    LineBuffer_3_Data};
        end
    endcase
end

always @(*) begin
    case(currentRdLineBuffer)
        0:begin
            LineBuffer_Read_Data[0] = Read_Line_Buffer;
            LineBuffer_Read_Data[1] = Read_Line_Buffer;
```

```
                LineBuffer_Read_Data[2] = Read_Line_Buffer;
                LineBuffer_Read_Data[3] = 1'b0;
            end
           1:begin
                LineBuffer_Read_Data[0] = 1'b0;
                LineBuffer_Read_Data[1] = Read_Line_Buffer;
                LineBuffer_Read_Data[2] = Read_Line_Buffer;
                LineBuffer_Read_Data[3] = Read_Line_Buffer;
            end
           2:begin
                 LineBuffer_Read_Data[0] = Read_Line_Buffer;
                 LineBuffer_Read_Data[1] = 1'b0;
                 LineBuffer_Read_Data[2] = Read_Line_Buffer;
                 LineBuffer_Read_Data[3] = Read_Line_Buffer;
           end
          3:begin
                 LineBuffer_Read_Data[0] = Read_Line_Buffer;
                 LineBuffer_Read_Data[1] = Read_Line_Buffer;
                 LineBuffer_Read_Data[2] = 1'b0;
                 LineBuffer_Read_Data[3] = Read_Line_Buffer;
           end
       endcase
end

lineBuffer lB0(
    .i_clk(i_clk),
    .i_rst(i_rst),
    .i_data(i_pixel_data),
    .i_data_valid(LineBuffer_Data_Valid[0]),
    .o_data(LineBuffer_0_Data),
    .i_rd_data(LineBuffer_Read_Data[0])
 );

 lineBuffer lB1(
     .i_clk(i_clk),
     .i_rst(i_rst),
     .i_data(i_pixel_data),
     .i_data_valid(LineBuffer_Data_Valid[1]),
     .o_data(LineBuffer_1_Data),
     .i_rd_data(LineBuffer_Read_Data[1])
  );

  lineBuffer lB2(
          .i_clk(i_clk),
          .i_rst(i_rst),
          .i_data(i_pixel_data),
          .i_data_valid(LineBuffer_Data_Valid[2]),
          .o_data(LineBuffer_2_Data),
          .i_rd_data(LineBuffer_Read_Data[2])
       );

       lineBuffer lB3(
           .i_clk(i_clk),
           .i_rst(i_rst),
           .i_data(i_pixel_data),
           .i_data_valid(LineBuffer_Data_Valid[3]),
           .o_data(LineBuffer_3_Data),
           .i_rd_data(LineBuffer_Read_Data[3])
    );

endmodule
```

B.2 line Buffer

```
‘timescale 1ns / 1ps
module Line_Buffer(
input    i_clk,
input    i_rst,
input  [7:0] i_data,
input    i_data_valid,
output  [23:0] o_data,
input i_rd_data
);
```

```
reg [7:0] line_Mem [511:0];
reg [8:0] Write_Pointer;
reg [8:0] Read_Pointer;

assign o_data = {line_Mem[Read_Pointer],
                line_Mem[Read_Pointer+1],line_Mem[Read_Pointer+2]};

always @(posedge i_clk) begin
    if(i_rst) begin
        Write_Pointer <= 'd0;
        Read_Pointer <= 'd0;
    end
    else if(i_data_valid) begin
        line[Write_Pointer] <= i_data;
        Write_Pointer <= Write_Pointer + 'd1;
    end
    if(i_rd_data)
        Read_Pointer <= Read_Pointer + 'd1;
end

endmodule
```

B.3 Convolution block

```
`timescale 1ns / 1ps
module Convolution_Block(
input         i_clk,
input  [71:0] i_pixel_data,
input         i_pixel_data_valid,
output reg [7:0] o_convolved_data,
output reg    o_Convolved_Data_Valid
    );

integer i;
reg [7:0] Kernel_1 [8:0];
reg [7:0] Kernel_2 [8:0];
reg [10:0] Mult_Data_1[8:0];
reg [10:0] Mult_Data_2[8:0];
reg [10:0] Sum_Data_Int_1;
reg [10:0] Sum_Data_Int_2;
reg [10:0] Sum_Data_1;
reg [10:0] Sum_Data_2;
reg Mult_Data_Valid;
reg Sum_Data_Valid;
reg Convolved_Data_Valid;
reg [20:0] Convolved_Data_Int_1;
reg [20:0] Convolved_Data_Int_2;
wire [21:0] convolved_data_int;
reg Convolved_Data_Int_Valid;

initial begin
    Kernel_1[0] = 1;     Kernel_1[1] = 0;     Kernel_1[2] = -1;
    Kernel_1[3] = 2;     Kernel_1[4] = 0;     Kernel_1[5] = -2;
    Kernel_1[6] = 1;     Kernel_1[7] = 0;     Kernel_1[8] = -1;

    Kernel_2[0] = 1;     Kernel_2[1] = 2;     Kernel_2[2] = 1;
    Kernel_2[3] = 0;     Kernel_2[4] = 0;     Kernel_2[5] = 0;
    Kernel_2[6] = -1;     Kernel_2[7] = -2;     Kernel_2[8] = -1;
end

always @(posedge i_clk) begin
    for(i=0;i<9;i=i+1)      begin
        Mult_Data_1[i] <= $signed(Kernel_1[i])*
                          $signed({1'b0,i_pixel_data[i*8+:8]});
        Mult_Data_2[i] <= $signed(Kernel_2[i])*
                          $signed({1'b0,i_pixel_data[i*8+:8]});
    end
    Mult_Data_Valid <= i_pixel_data_valid;
end


always @(*) begin
    Sum_Data_Int_1 = 0;
    Sum_Data_Int_2 = 0;
    for(i=0;i<9;i=i+1)      begin
        Sum_Data_Int_1 = $signed(Sum_Data_Int_1) +
                        $signed(Mult_Data_1[i]);
        Sum_Data_Int_2 = $signed(Sum_Data_Int_2) +
                        $signed(Mult_Data_2[i]);
    end
end

always @(posedge i_clk) begin
    Sum_Data_1 <= Sum_Data_Int_1;
    Sum_Data_2 <= Sum_Data_Int_2;
    Sum_Data_Valid <= Mult_Data_Valid;
end

always @(posedge i_clk) begin
    Convolved_Data_Int_1 <= $signed(Sum_Data_1)*$signed(Sum_Data_1);
    Convolved_Data_Int_2 <= $signed(Sum_Data_2)*$signed(Sum_Data_2);
    Convolved_Data_Int_Valid <= Sum_Data_Valid;
end

assign convolved_data_int = Convolved_Data_Int_1+
                Convolved_Data_Int_2;

always @(posedge i_clk) begin
    if(convolved_data_int > 4000)
        o_convolved_data <= 8'hff;
    else
        o_convolved_data <= 8'h00;
    o_Convolved_Data_Valid <= Convolved_Data_Int_Valid;
end

endmodule
```

B.4 Top level block integration

```
'timescale 1ns / 1ps
module Image_Processing_Module(
input    axi_clk,
input    axi_reset_n,
input    i_data_valid,
input  [7:0] i_data,
output   o_data_ready,
output   o_data_valid,
output [7:0] o_data,
input    i_data_ready,
output   o_intr

    );

wire [71:0] pixel_data;
wire pixel_data_valid;
wire axis_prog_full;
wire [7:0] convolved_data;
wire convolved_data_valid;

assign o_data_ready = !axis_prog_full;

Control_Circuit IC(
    .i_clk(axi_clk),
    .i_rst(!axi_reset_n),
    .i_pixel_data(i_data),
    .i_pixel_data_valid(i_data_valid),
    .o_pixel_data(pixel_data),
    .o_pixel_data_valid(pixel_data_valid),
    .o_intr(o_intr)
  );

 Convolution_Block conv(
     .i_clk(axi_clk),
     .i_pixel_data(pixel_data),
     .i_pixel_data_valid(pixel_data_valid),
     .o_convolved_data(convolved_data),
     .o_convolved_data_valid(convolved_data_valid)
 );
```

```
  outputBuffer OB (
    .wr_rst_busy(),
    .rd_rst_busy(),
    .s_aclk(axi_clk),
    .s_aresetn(axi_reset_n),
    .s_axis_tvalid(convolved_data_valid),
    .s_axis_tready(),
    .s_axis_tdata(convolved_data),
    .m_axis_tvalid(o_data_valid),
    .m_axis_tready(i_data_ready),
    .m_axis_tdata(o_data),
    .axis_prog_full(axis_prog_full)
  );


endmodule
```