An Architectural Design For Gaussian Filter

Aditya Raj Singh Gour (19116005)
Gopal Gupta (19116025)
Hardik Rathore (19116028)
Rudraksh Aggarwal (19116061)

Abstract—The Gaussian Filter is one of the processing methods to smoothen images. Before detecting an object, this method is used as a pre-processing step, for eliminating useless detail, and connecting unnatural parts. Gaussian filter is a low pass linear filter as it reduces high frequency components of the image. Mathematically, applying a Gaussian filter to an image is the same as convolving the image with a Gaussian kernel. In this project the image is being converted to a .hex file (used Python for the purpose) initially then after applying the Gaussian convolution to each pixel of the image (used Verilog for this purpose) it is again converted back to the image format.

Keywords—Gaussian Filter, Kernel, Convolution

I. INTRODUCTION

This paper proposed the implementation of Gaussian Filter method for the pre-processing of the image.

The Gaussian filter is used to blur an image in order to reduce the high frequency components. It is an average and low pass filter. It replaces the pixel by the weighted mean of the neighboring pixel intensity thus giving more importance to the central pixel and less to distant ones. This process takes place by Gaussian Convolution.

Gaussian filter is one of the most widely used filters and is of great interest in computer vision and image preprocessing. It helps to process the image by removing the unnecessary details and connecting different absurd parts of the image. It is of prime use in edge detection of an image.

Implementation of Gaussian filter involves

(1) Converting the image file to .hex file.

(2) Converting the .hex file into a matrix where each element corresponds to a pixel.

(3) Determining the size of the kernel and fixing it.

(4) Applying Gaussian convolution on the image matrix with the help of the kernel.

(5) Storing the output values of step (4) in an output matrix.

(6) Converting the output matrix into the output image file.

II. ARCHITECTURE OF THE DESIGN

Initially it was planned to make a hardware design, but due to unavailability of resources and ongoing Covid-19 pandemic, we were not able to work on any hardware related stuff. Hence, the project was limited to just implementation of the software Component(Verilog Code).

Following is the detailed procedure that takes place for applying Gaussian filters on the image.

First of all, two different testbenches were made for this project one for applying Gaussian filter to Colored images(testbench_color.v) and other for applying filter on Grayscale images(testbench.v).

(1) We need to provide the modules of the test bench with the input parameters - the input image , the kernel matrix order an integer in the range of {3,5,7}, rows of image, columns of image and size of image.
(2) Now this input image is required to be in a .hex file. So it is needed to convert the image file to .hex. For this two codes were written one in MATLAB for converting .bmp file to .hex and another in PYTHON for converting .jpg file to .hex.(NOTE: Make sure that image is need to be Grayscale for using testbench.v)
(3) The structure of the module called by our testbench is described below:
● First a function (Verilog $readmemh() property) is called that converts the .hex file to a Rectangular Matrix of

order M X N (passed as a row and column parameter by our testbench)

- Then from the integer input received ,by using one of the self made functions it designs a Kernel(using Verilog functions and loops).
- Once the Kernel and Matrix are made the module performs the processing of the image matrix (backbone of the project) i.e. CONVOLUTION using another function, which finally gives an output matrix of order M X N.
- Finally this output matrix is converted to a .hex file for further use and marking the end of our verilog code.

Now, from this .hex file the new output image is generated using PYTHON.

Also, to test the results a gaussian filter was directly applied on the same image using one of the functions in PYTHON's inbuilt library.
The new image is the same image as input but with the Gaussian Filter applied on it i.e. a much smoother image than the input which is all set to be used for further processing.

III. CHALLENGES FACED DURING IMPLEMENTATION

We faced a few problems while working on the project:

(1) Processing of Edge pixels, there were majorly three different approaches for applying gaussian filter on edge pixels while convolution each having its own demerit. So we went for the one we felt was the best.

(2) Our first few stimulation test programs were hanging and after some debugging we realized that the code went into an infinite loop and fixed the same.

(3) We wanted to take an input for the order of our Gaussian kernel at run time but were unable to find if it's possible. Hence we decided on an alternative where the user could pass a parameter in the test bench called ksize while calling the module named process which will be the order of the kernel

(4) Currently, we have to send the image converted to hex file to verilog code manually and then the generated hex file by verilog to python for conversion to output image manually too. We tried automating this process but failed to do so.

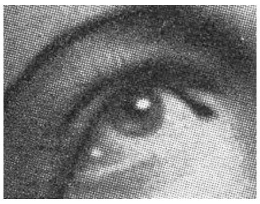
We also faced some problem due to the ongoing Covid-19 pandemic :

- Hardware implementation of the project was not possible.
- Lack of proper guidance in person also affected the project.Also faced problems while communicating with team members.

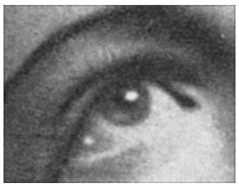
IV. SIMULATION RESULTS

We passed the Input Image to the code which after being processed by the Gaussian Filter gave an Output Image of the same dimensions as follows. We processed the Input Image using a Kernel of size 3, 5 and 7. We also processed both grayscale and colored images as input which gave satisfying results with a slight variation in the code,

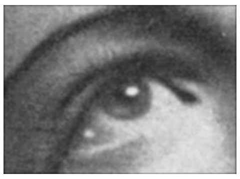
Input Image:



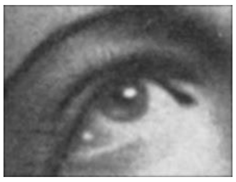
Output Image:



(Order 3 Kernel)



(Order 5 Kernel)



(Order 7 Kernel)

For Grayscale Images For Coloured Images

V. CONCLUSIONS

The paper here proposed the architectural design of Gaussian filter for the preprocessing or an image. The implementation of the filter proposed correct results for the Kernel size 3, 5 and 7 (It gave the same results when the Gaussian filter was applied on the image using Python) . We now look forward to the Hardware implementation of the filter in the real time whenever possible.

(4)

(5) ACKNOWLEDGEMENT

This project was a part of the course curriculum of ECN-104 (Digital Logic design) taught by Proff. Bishnu Das. It was completed under his supervision and we thank him for helping us throughout the project . We would also like to acknowledge the invaluable support from our seniors from the institute.

(6) REFERENCES

[1] Summin Song, SangJun Lee, jae Pil Ko, Jae Wook Jeon "A Hardware Architecture Design for Real-time Gaussian Filter " IEEE International Conference on Industrial Technology (ICIT), Feb. 26 - Mar. 1, 2014, Busan, Korea

[2] Image Processing on FPGA using Verilog HDL fpga4student.com

[3] Blogs from medium.com

[4] Docs.Opencv.org (documentation on filters and image processing)

[5] NPTEL Lecture by Indrail Sengupta IIT Kharagpur

[6] Project resources provided by Proff. Bishnu Das