CS203 PROJECT REPORT

TOPIC – IMAGE PROCESSING

DONE BY –

- **ABHIJITH T R – 2020CSB1062**
- **ADITYA AGGARWAL – 2020CSB1066**

INSTRUCTOR

- **DR NEERAJ GOEL**

ABSTRACT

In our project, we seek to gain an understanding of image representation in computers and seek to study the various methods available to read them through scripts. We seek to gain an understanding of the various simple image manipulation operations and the methods available to perform these operations in Verilog.

INTRODUCTION

Image processing is an extremely useful tool which is already in use in several domains around us, such as face recognition, medical diagnosis, entertainment and computer vision. Our project aims to perform simple image manipulation operations on various images.

Image processing aims to change the nature of images or in some cases to extract useful data from the images. In our project, we have done the former. Using Verilog code, we extract the data of the image pixel-by-pixel and perform operations like brightness control, inversion and grayscaling.

We seek to understand various properties of images stored in the pixel data and how this pixel data is stored in an ordered manner leading to the formation of images in hex files and bmp files and how these pixel values can be manipulated to generate different versions of the same image.

IMAGE PROCESSING USING VERILOG

In this section, we seek to provide a brief overview of the entire project before moving onto the manipulations required to perform individual operations in later sections. Individual Verilog modules have been created for each image processing operations. These can be run individually on the terminal if only a single operation is to be performed.

The Verilog scripts will copy the hex data from the .hex files into registers which will then be used for manipulation. A new register will be used to store the final data which will be written into a new .hex file in the end using Verilog file write functions.

The Verilog scripts perform the operations on a hex file which is present in the working directory and generate another hex file as the output. The

conversions from bmp format to hex format is being controlled by external MATLAB scripts. The user has to interact with the python script which will give the user various options and ability to change the image to be processed.

ALGORITHMS USED

In this section, we seek to provide an understanding of how the various conversions were achieved once the required pixel data was available to us. In some operations it is important to realise that 255 takes the image towards white and 0 takes the image towards black.

1. Brightness control

The image data was stored in a 2D register that consisted of m × n locations each of 8 bits to store the pixel values from 0 to 255. Once the image data is available, to increase the brightness, we add 32 to the individual pixel values and to decrease the brightness we simply multiply the pixels by 0.5, while taking into account that the pixel values cannot be greater than 255 or less than 0.

Original Image

Brightness Decrease

Brightness Increase

2. Flip and mirror

The image data is stored in a 2D register. We simply swap the mirror values at either ends i.e., we simply swap 1 with size-1, 2 with size-2 and so on. This leads to an overall flip as well as mirror at the same time.

Original Image

Mirror & Flip Operation

3. Grayscale & Mirror

In order to convert to grayscale, we need to set the red, green and blue components of each pixel to the same value. We do this by a very simple method. We simply add the R, G and B components of each pixel and take its average and assign this average to all 3 components. We perform mirror operation by swapping pixels (indexed from 0) in a row that sum up to number of columns-1.

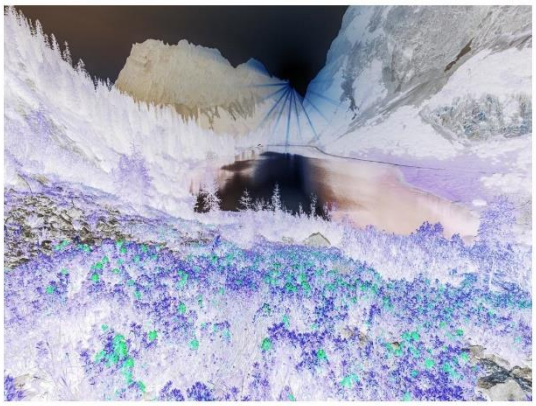
Original Image

Grayscale & Mirror Operation

4. Inversion

The conversion to an inverted image follows a very simple method. We simply take every single pixel and convert the R, G and B components to 255-R, 255-G and 255-B respectively.

Original Image



Inversion Operation

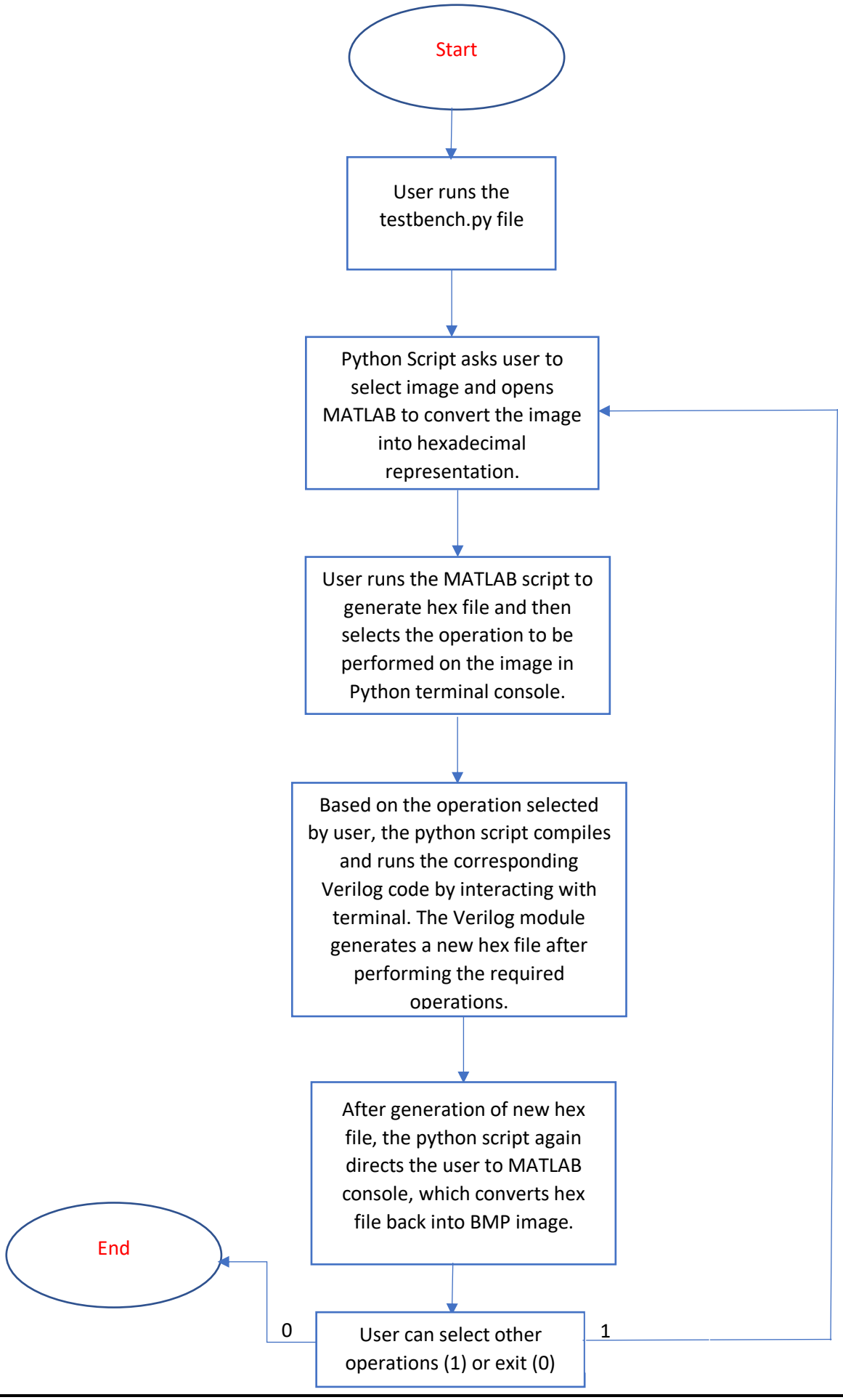
5. Threshold

The threshold image consists of only two colours i.e., black and white. In our particular implementation, the decision has been taken at an arbitrary sum value of the R, G and B components which is 330. If the sum of the three components is greater than this value, the pixel is set to (255, 255, 255) to signal white, else it is set to (0, 0, 0) to signal black. (R, G and B values in parentheses)

Original Image

Threshold Operation

Flowchart Description of Programme Behaviour



CONCLUSION

We have studied the representation of image pixel values in terms of their red, green and blue components in .hex files and have implemented simple MATLAB scripts to convert from .hex to .bmp for viewing. We have studied simple image processing operations which can be scaled up or applied in parallel to perform much more complicated operations. We have implemented simple Verilog scripts to achieve the required operations that can be run independently.

Future developments of the project could be :

- Addition of more simple operations like sepia.
- Improvement of modules to allow multiple operations to work in succession.
- Research on advanced algorithms like Sobel Edge Detection and its implementation in Verilog.
- Real Time Image Processing on an FPGA board for various uses like medical surgeries, iris movement recognition etc.

ACKNOWLEDGEMENTS

We wish to thank our instructor Dr Neeraj Goel and all our teaching assistants for their guidance and valuable inputs provided during the building of the project.

REFERENCES

1. Wikipedia, *Digital Image Processing*, 2005
2. allaboutcircuits.com , *Digital Image Processing: Point operations to Adjust Brightness and Contrast*, 2019