METU EE314 Laboratory Spring 2016-2017 Term Project Coin Counter

Berk İSKENDER, Asena M. SARICI,

2031920, *berk.iskender@metu.edu.tr*, 2031284, *sarici.asena@gmail.com*

Abstract_This document describes the research conducted for the EE 314 Digital Laboratory Term Project and the method which will be used for the solution design of the given problem.

I. INTRODUCTION

In this term project, we are required to design a coin counter using Verilog and FPGA boards. We are supposed to use an input to FPGA, a 2D Grayscaled image of coins on a white sheet of paper marked with black squares of 1 cm edges. This document is composed of proposed solution, image loading and filtering techniques that are used for the detection of the circular shaped object in the image. Also, codes, block diagrams and images explaining the working principles of the theoretical background and outputs of the algorithms are included. These algorithms focus on taking the grayscale image as a .hex input, applying threshold filter, doing edge detection using 3x3 Sobel mask, CHT (Circular Hough Transform) for detecting the circles and giving the output of this process to the VGA display. However, since the preliminary report is based on the filtering techniques and theoretical background related to them, VGA partis not included and left for the final report. The overall block diagram of the project is shown in Figure 1.

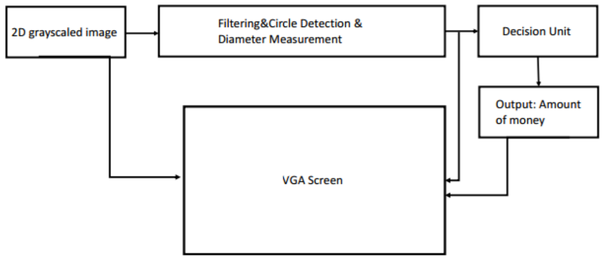


Fig 1. Block Diagram of the overall Project

II. PROPOSED SOLUTION

As a solution for the given problem, a detailed research on filtering techniques, uploading an image to FPGA [1], circle and edge detection algorithms, MATLAB built in codes for simple filter construction is conducted. We decided to use MATLAB for converting the image into a suitable form such that it is readable by FPGA. Then a filter design is seed suitable for the uploaded file. The next step is proposed as finding circles fitting into a range of diameter and counting their numbers using circle and square detection such that square pixel length will give a proportionality with the real dimensions of the image and the coins and counting process can be performed afterwards. Meanwhile, edge filtered array line is back converted to a file such that MATLAB can get the file and convert the pixel values written in the file back to image and this image is observable in the VGA screen. For this, it is proposed to use a suitable code later on in FPGA such that the monitor will understand where to stop and start a new line while showing the image. During the project, programming tools like MATLAB and Quartus, VGA interface, different images of coins and some other objects for filtering will be used.

III. IMAGE UPLOADING TO FPGA

Figure 2. depicts the process for uploading an image to FPGA, which is an important step for starting the project. The process is also as described in [1].

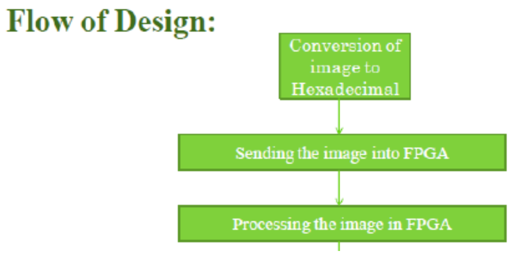
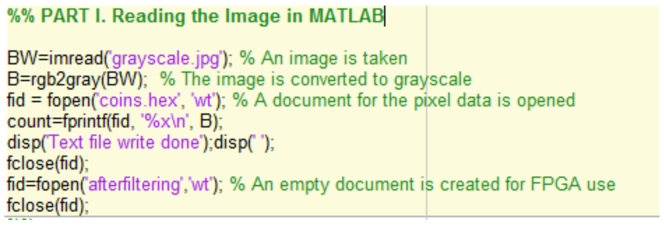

Flow of Design:
Conversion of image to Hexadecimal
Sending the image into FPGA
Processing the image in FPGA

Fig 2. Design flow for the Image Uploading [1]

The code shown in Figure 5 is used for reading the image and converting it to grayscale before writing the pixel values into a hex document for FPGA use. This step is required because creating a 320*240 matrix for FPGA in verilog is not useful in practice. Instead of using a 320*240 2D matrix, pixel entries are converted to a 1D matrix and transferred into a register having total number of elements equal to the total pixel number. After this transformation, image processing can be done on the FPGA, knowing the total number of rows and columns that the image contains beforehand and preparing the remainşng algorithm accordingly.



```
%% PART I. Reading the Image in MATLAB

BW=imread('grayscale.jpg'); % An image is taken
B=rgb2gray(BW);  % The image is converted to grayscale
fid = fopen('coins.hex', 'wt'); % A document for the pixel data is opened
count=fprintf(fid, '%x\n', B);
disp('Text file write done');disp(' ');
fclose(fid);
fid=fopen('afterfiltering','wt'); % An empty document is created for FPGA use
fclose(fid);
```

Fig 3. MATLAB code used for Reading the Image

IV. FILTERING TECHNIQUES

A. Edge Filtering Algortihms

For edge filtering, high contrast image points can be found by intensity difference computations in the images. These areas of high intensity form the borders of different objects using the surrounding masks which are matrices representing some derivative operations. These masks are multiplied by the image vectors so that the difference at each point with respect to its surrounding is obtained.

There are different types of masks that can be used for edge detection. These include Prewitt, Sobel which are 3 by 3 masks and Roberts which is a 2 by 2 mask. The difference between Sobel and Prewitt is that the center estimate of Sobel is twice more intense. The matrices are shown clearly in Figure 4.

$$\textbf{Prewitt:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \;;\quad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$$

$$\textbf{Sobel:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \;;\quad M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

$$\textbf{Roberts:}\quad M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array} \;;\quad M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}$$

Fig 4. Masks Used in Image Filtering [2]

$$\textbf{gradient:}\quad \mathbf{W_1} = 1/\sqrt{8}\begin{array}{|c|c|c|}\hline 1 & \sqrt{2} & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -\sqrt{2} & -1 \\ \hline\end{array} \qquad \mathbf{W_2} = 1/\sqrt{8}\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \sqrt{2} & 0 & -\sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline\end{array}$$

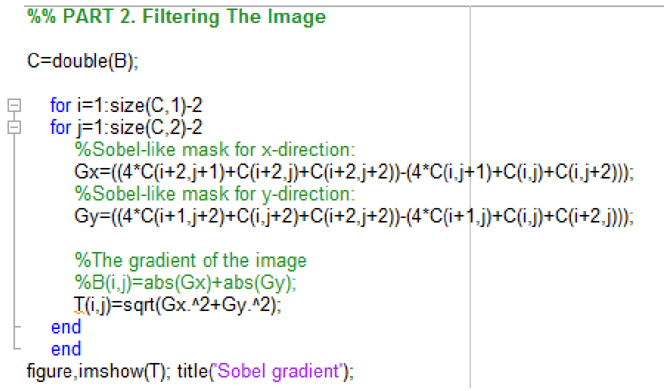
Fig 5.Gradient Type for Different Type of Filter Usage[2]

Another filtering technique is Canny Edge Detector which first smooths the intensity of the image and after that highlights the contours.

After some research, it is observed that masks are the basics for the filtering operation. The response of a mask to a certain region in an image is proportional to how similar that neighbourhood looks like to the mask.[2]. We can design a mask for tasks we want to perform like edge detection or special pattern detection like corners or circles etc. Usually masks used for image processing are 3x3 or higher and the elements in the sets need to be orthogonal.

In our filter design, we take Sobel Filter Design as base and using the gradient technique showed in Figure 5 we propose a filter with element values composed of 1's and 4's instead of 1's and 2's in Sobel Filter. This way we aim to strengthen the difference of the boundaries in the image. The matrix can be divided to its weight as proposed in Figure 5 to keep the intensity constant but we do not consider it here. Moreover the 3x3 choice is for the simplicity of the implementation on FPGA for later use of the filter. The mask used for

Then the rest of the procedure will be conducted on FPGA. However for the ease the filter is first tested on MATLAB using the code depicted in Figure 6. Then another filtering is applied for threshold to see the edges as clear as possible. The code used fort his procedure is provided in Figure 7. After these steps, as a result we obtained the filtered image as in Figure 8.a.



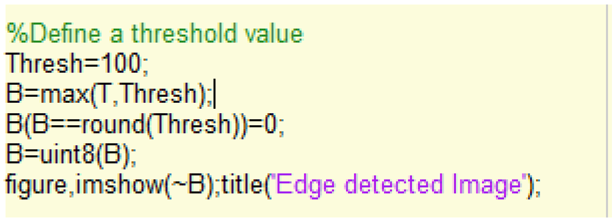
```
%% PART 2. Filtering The Image

C=double(B);

for i=1:size(C,1)-2
  for j=1:size(C,2)-2
      %Sobel-like mask for x-direction:
      Gx=((4*C(i+2,j+1)+C(i+2,j)+C(i+2,j+2))-(4*C(i,j+1)+C(i,j)+C(i,j+2)));
      %Sobel-like mask for y-direction:
      Gy=((4*C(i+1,j+2)+C(i,j+2)+C(i+2,j+2))-(4*C(i+1,j)+C(i,j)+C(i+2,j)));

      %The gradient of the image
      %B(i,j)=abs(Gx)+abs(Gy);
      T(i,j)=sqrt(Gx.^2+Gy.^2);
  end
end
figure,imshow(T); title('Sobel gradient');
```

Fig 6. MATLAB code used for Filtering the Image



```
%Define a threshold value
Thresh=100;
B=max(T,Thresh);
B(B==round(Thresh))=0;
B=uint8(B);
figure,imshow(~B);title('Edge detected Image');
```

Fig 7. Threshold code

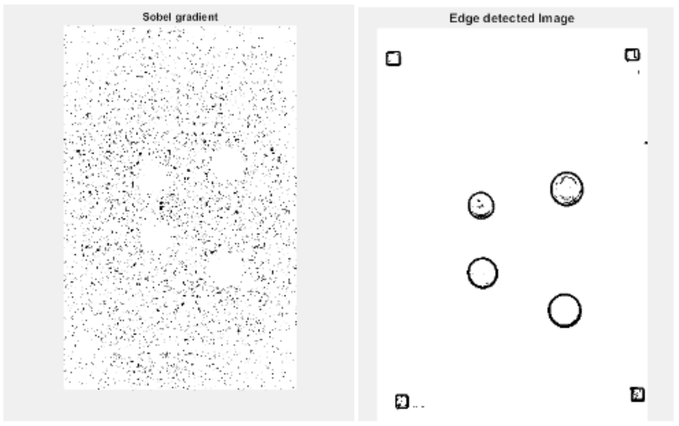


Fig. 8.a On left, the filtered image before thresholding. 6.b On right, filtered image after thresholding

After the edge detection procedure for circle detection Hough Transform is used.

B. *Hough Transform for Circle Detection*

Hough transform is utilized for detecting analytical shapes in an image. For our purpose it will be used for detecting circles which have radii lying in a specified interval representing different coins. Then using accumulation, the number of circles in a radius class is counted. This will give us the total amount of money on the grayscaled picture. For this type of filter, a grayscale image should be used as an input and to avoid unnecessary circle detections which do not correspond to real circular objects, a threshold mask and an edge detection mask can be used. For this purpose we used Sobel filter. This transform has an algorithm that aims to detect if an edge exists for that particular pixel value. If that pixel is proved to be an edge, its parameters are measured and related variable for that candidate parameter is increased. In this procedure, voting algorithm is implemented. Local maxima corresponding to parameters are found which are "voted" most by the algorithm and used in the transform. This maxima is found in the accumulator space which is described by the circle expression in the 2D space. [3]

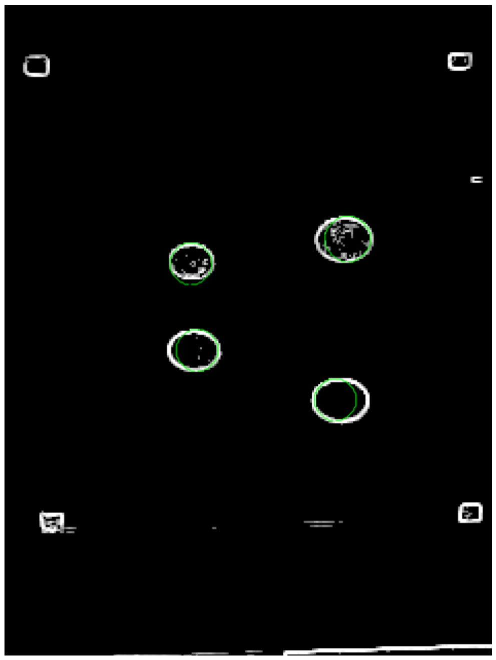


Fig 9. Circle detection using Hough transform in MATLAB

B2. *Another candidate method for circle detection*

Due to the difficulties of implementation of Circular Hough Transform algorithm on FPGA, another method which utilizes the total number of black or white pixels (depending on the threshold technique) can be used. In this method, since the coins are circular, image is investigated horizontally. After the detection of first different pixel (corresponding to an object on the paper) subsequent different colored pixels are counted vertically and their midpoint can be found. After finding the midpoint, a horizontal count is done on the object. If their sum is close to each other (vertical radius and horizontal radius) with a threshold that is determined beforehand, (e.g total difference of 8 pixels) the object can be interpreted as a circle. After the detection, minimum square region found by the algorithm which includes the circular shaped object can be attained new pixel values as equal to the font (paper) to prevent new detections for the same coin in further iterations.

V. Conclusion

In this preliminary report, methodology of a general circle detection is examined and related research is done. For this purpose, common algorithms are investigated and feasible ones are chosen for further implementation on the environment that will be used in the upcoming parts of the project. Initially, grayscaled image is turned into an .hex file which includes the pixel values as one dimensional. After this, a threshold filter which makes pixels black or white by comparing them with a reasonable value is applied to increase the difference on the edges of the objects in the image. Then, a 3x3 Sobel mask is used to compute gradients on the edges and depending on the magnitudes of gradients, edges are obtained by comparison. After edge detection, circular hough transform is utilized for circle detection which makes use of the voting system and an accumulator for a range of parameters and detects the maxima corresponding to a circle with that parameters by marking it on the object.

References

[1] Dhanabal.R, Sarat kumar Sahoo, Bharathi V, "FPGA BASED IMAGE PROCESSING UNIT USAGE IN COIN DETECTION AND COUNTING" in *International Conference on Circuit, Power and Computing Technologies [ICCPCT]*, 2015.

[2] L.G. Shapiro and G.Stockman, "Filtering and Enhancing Images," in Computer Vision, Mar 2010.

[3] J. Borovicka, "Circle Detection Using Hough Transforms Documentation COMS30121 - Image Processing and Computer Vision"

[4] Edge Detection on Images Available at :https://www.mathworks.com/help/coder/examples/edge-detection-on-images.html?prodcode=ML. [Online] Accessed: May 20, 2017.

[5] Hough transform for circles[Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/26978-hough-transform-for-circles: May 20, 2017.