**Edge Detection Using an FPGA**

by Aidan Blant, Yinyi Chen, Emma Smith

**Abstract**

We are trying to create a better solution for edge detection of images when using an FPGA for parallel processing. By focusing on how we split up the image input and storage, the number of parallel processes, and the method of thresholding we will be able to create a project that works quickly and produces a valid detected edges image.

**Introduction and motivation**

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting points in the image where there is a sharp contrast in brightness, and strings the points together to form edge segments. Edge detection a fundamental step for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

One of the main challenges we will face is to optimize the use of the Nexys board and its ability to run parallel processes. By decomposing the Sobel algorithm and identifying how is best to split up the image we can improve the speed at which our process runs as well as use less total number of operations. The image is broken down into as many parts as there will be processes which will perform the Sobel algorithm on each pixel using 3x3 kernels. Then as we stitch together these parts into a full image, we must consider the borders between them. We will look at two possible approaches, initially adding on a border to each part in the image to be processed in the parallelized portion of the algorithm or leaving the border meshing to the later single process stages.

Another challenge faced in edge detection is the ability for the algorithm to work on any image, regardless of the number of edges or the amount of contrast in the image. There is a threshold value the algorithm uses to decide if a particular point should be considered part of an edge and it greatly impacts the final output. We want to implement a method to derive a threshold value for each image such that varied inputs could be used with good results, rather than using a hardcoded value.

**Previous work**

Edge detection is a well know problem with many applications, so there is much prior work on the topic with various aspects specifically targeted. From our research, we have found that FPGAs have been used for their large memory which provides a platform for processing of the Sobel algorithm with substantially higher performance than programmable digital signal processors.[[1]](#footnote-1) The fact that our Nexys board has a fairly substantial amount of memory means that we can read in the entire image to bits and store in memory at once to work on with multiple processes. There was a paper that used soft-threshold wavelets to remove noise of the picture when applying the Sobel operator resulting in a better picture output.[[2]](#footnote-2) As well as a paper that utilized pattern-recognition and probabilities in testing the enhancement of thresholding for various images.[[3]](#footnote-3) While we are unsure as of yet what exact method we will use to optimize the threshold, but these techniques can be explored along with a few other ideas we have come up with.

**Approach and concrete experiments**

To create an improved version of the Sobel edge detection method, we need to maximize the ability of our process to correctly detect edges while also focusing on the speed of execution. To complete the edge detection algorithm on the Nexys board we created a stateflow diagram where we can parallelize certain parts and will have to optimize the single process sub-modules.

The first stage is prepping the images by converting to a grayscale 2D matrix and partitioning them into the number of parallel processes used in the next steps. In this step we are writing the data to memory and generating the surrounding matrix for each pixel. Next we actually apply the Sobel kernel filter via convolution to each matrix resulting in a gradient and magnitude of the possible edge. The last step which we can parallelize is the filtering, where we compare the values to the threshold and assign a new pixel value to that spot after determining if it will be considered an edge. Finally, we fill the full size matrix with the new pixel values that indicate where edges exist and output it.

Currently, we have a functioning stateflow for one single or multiple processes to execute at the same time. For this milestone, we were focused on creating a working design and novel approach. Rather than reading in an image we were testing the validity of our design on a hard coded matrix so we could verify that the Sobel operations were outputting the correct matrix and properly dividing and recombining the matrix when in parallel.

**Conclusion and short-term plans**

Overall we have the basis of our project completed and understand what aspects of the process will affect our speed and output so that we can implement it in the optimal way. For Milestone 3, we would like to have the Nexys board taking in the input image, processing it, performing the Sobel operations, and outputting an image with the detected edges. Our future optimization tests will include running with a variation in the number of parallel processes and running with different threshold levels on a variety of pictures. When we have a full path working it will help us to then test various configurations to see what works quickest and most efficiently.

1. A novel FPGA-based architecture for Sobel edge detection operator

   T. A. Abbasi; M. U. Abbasi

   International Journal of Electronics [↑](#footnote-ref-1)
2. An improved Sobel edge detection

   Wenshuo Gao; Xiaoguang Zhang; Lei Yang; Huizhong Liu

   Computer Science and Information Technology [↑](#footnote-ref-2)
3. Quantitative design and evaluation of enhancement/thresholding edge detectors

   Abdou, I.E.; Pratt, W

   Proceedings of the IEEE [↑](#footnote-ref-3)