

ECE 480 – Design Team 3

Live Video Edge Detection in Simulink

**Chad Eckles
Dr. Ramakrishna Mukkamala**

**Application Notes
Friday November 19, 2010**

Executive Summary

While technology is incorporating more object detection and recognition into devices, the heart of all these systems starts in edge detection. Some of the various systems that use edge detection for object detection are facial recognition, lane detection warning, lost object detectors, and more. Automobile companies have been investing more in advancing systems to aid/improve automobiles for drivers. Xilinx has been a leader by using their FPGA systems to perform these tasks. Xilinx FPGA edge detection algorithms are created using the program Simulink. When first learning how to create the algorithms, a Simulink model must be developed to fully understand how edge detection works and how it can be broken down to later on model it in Xilinx blocks.

Keywords

Edge Detection, Xilinx, FPGA, Simulink, MATLAB, Video and Image Processing.

Table of Contents

Executive Summary and Keywords	1
Table of Contents	2
Introduction.....	3
Objective	3
Implementation and Results	4
Conclusion.....	9
References	9

Introduction

Our sponsor Xilinx has started an on going project, with this being the first year, on developing a FPGA driver assistance rear view camera system. After extensive amounts of research performed by our design team we found that the best starting place is to implement edge detection. Edge detection usually is the base for all other video processing because to perform any object detection or recognition first all the edges in the image have to be detected. Once all the edges are detected, the ones of non-importance can be filtered out. The edges of importance then can be used to detect where an object is. From there the object can be boxed or other techniques can be performed.

Since there is not an easy way to program an FPGA system, our team thought the best way would be to first create an algorithm in Simulink for edge detection to fully understand how the system works. Using this system we can learn how to convert it to a model for FPGA use. By modeling in Simulink we can understand how edge detection works on live video and how to perform other operations.

Objective

The purpose of this application notes is to familiarize people on Simulink with an emphasis on explaining how to implementing basic image and video processing techniques. Exposure with this program is usually limited and for this project, Simulink becomes a key component. Since the Xilinx FPGA project is an on going project it is best for future teams to have documents on how to properly get started with Simulink and video processing.

Implementation and Results

The creation of this design is in a sub-program of MATLAB, so MATLAB needs to be opened first. Once MATLAB is opened we then need to open Simulink. This can either be opened by typing Simulink in the command prompt on the MATLAB home screen or by clicking the Simulink icon at the top. If you look at figure 1 below, the top red circle is highlighting where the Simulink quick launch icon is and the red circle below that is where to type “Simulink”.

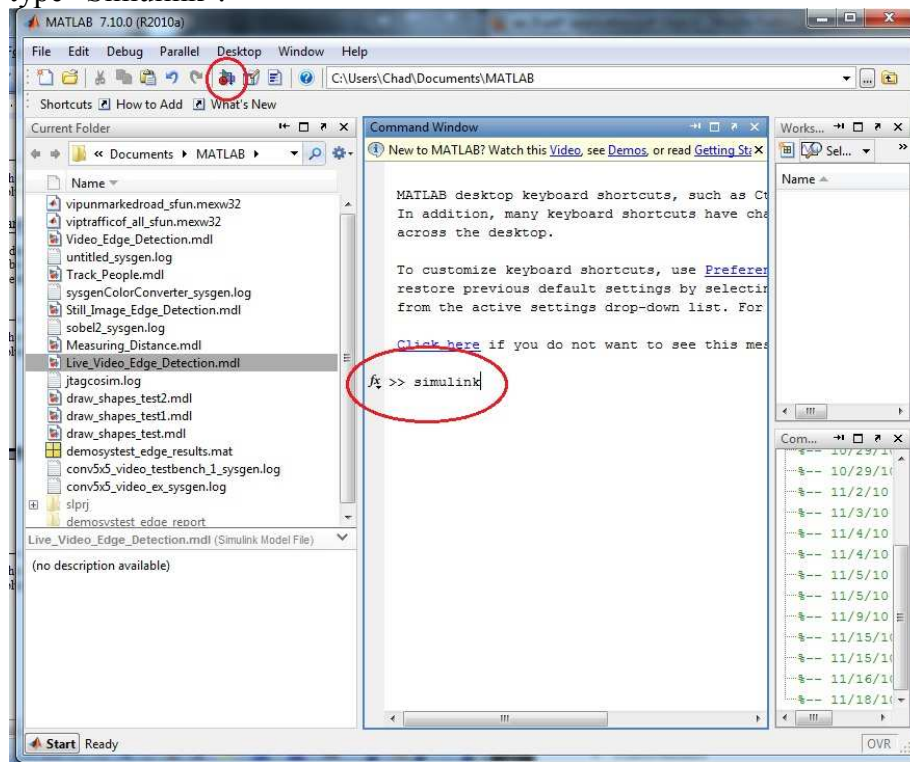


Figure 1

After opening Simulink, all of the block for building models will appear in the list. Next we need to open a new file to begin building our model. To accomplish just go to file then click new.

The best starting place is to get the input and output in our model, then build around them. Running edge detection on live video can be accomplished using a video off the hard drive or using another source like a webcam. For the majority of this application note it is going to be done using a webcam but I will mention how to read a video from the hard drive. For the input we are going to use a block called “From Video Device”, located in the Image Acquisition Toolbox. This block will find the appropriate device that will capture live video. The computer being used is equipped with a webcam so that will be the default device. The red circle in figure 2 is the library where the “From video Device” block can be located, while the blue circle is the block. To use a video from the hard drive, the block needed is “From Multimedia File”. This block is located in the sources library.

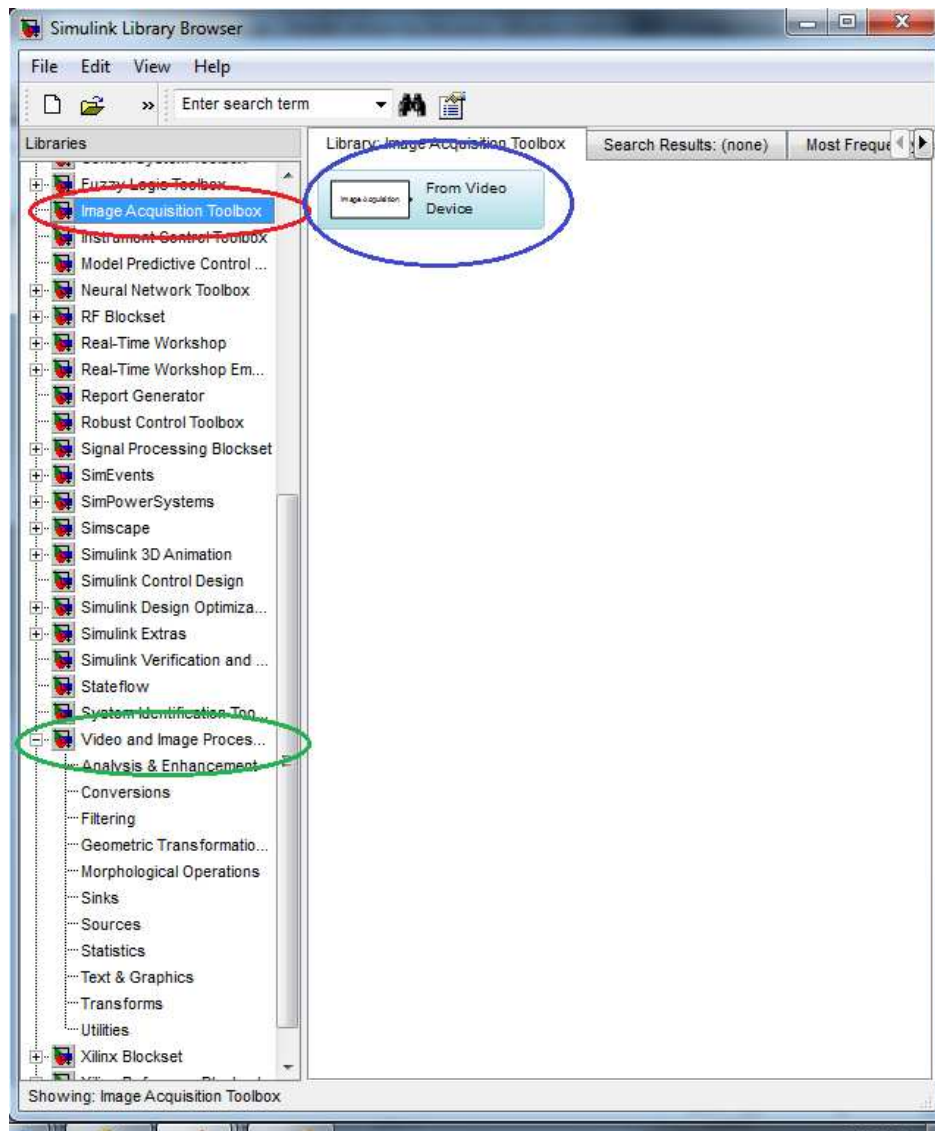


Figure 2

To get the “From Video Device” block into our design, click and hold on the block in the library, then drag it into the blank document. The next block we want is our output block. The block is called “To Video Display”. This will output our final video signal onto a viewable screen that will pop up once we run the model. The block is located in the video and image processing library. When looking at figure 2, this library is highlighted by a green circle. Most of the other blocks will be located in here. With the two blocks in our model it should look like figure 3.

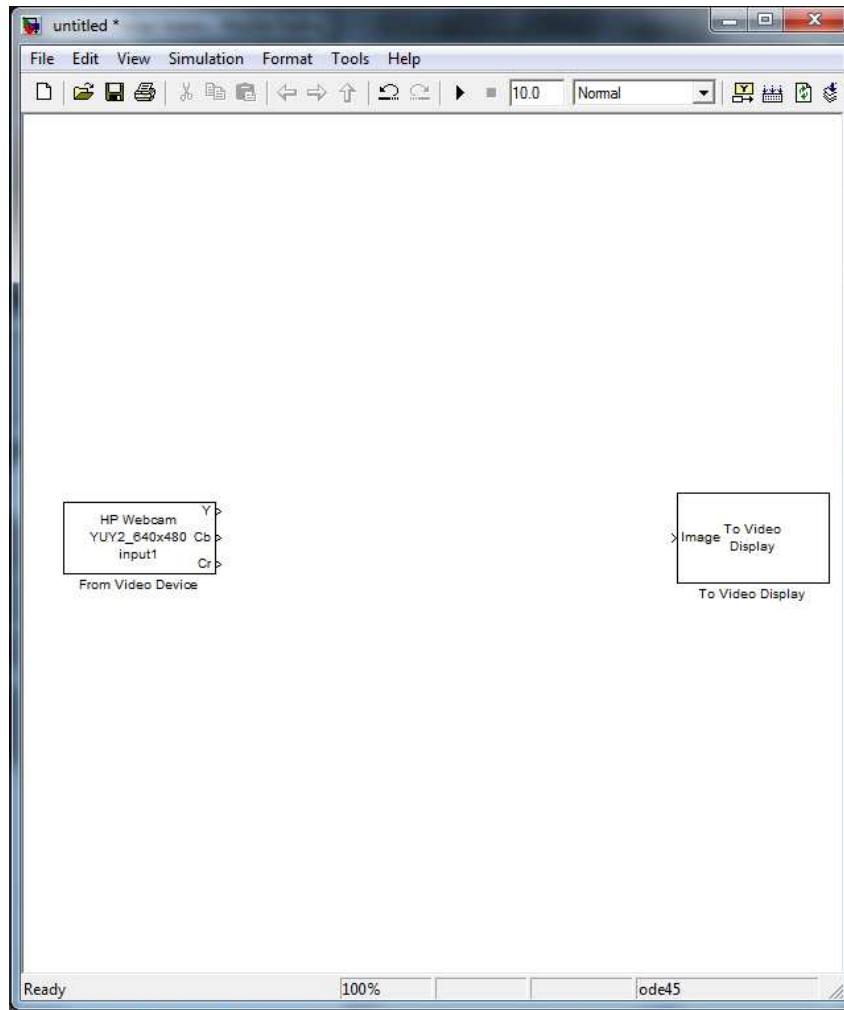


Figure 3

The “From Video Device” block needs to separate the output signals to Y, Cb, Cr like shown in figure 3. To accomplish this, double click on the block and change port mode from one multidimensional signal to separate color signals.

To implement edge detection, first the color video has to be converted to grayscale. To convert the video we have to first convert Y, Cb, Cr to R, G, B then convert R, G, B to intensity (intensity is grayscale). Both of these conversions can be done using two of the same blocks, which is called a color space conversion. This block is located in the conversions library, which can be seen in figure 2. Again we need to change all of these blocks to separate signals and that is accomplished like previously stated. The conversion type needs to be changed to the appropriate type. This is all accomplished in the mask parameters. All the connections and appropriate conversion types are located in figure 4 below.

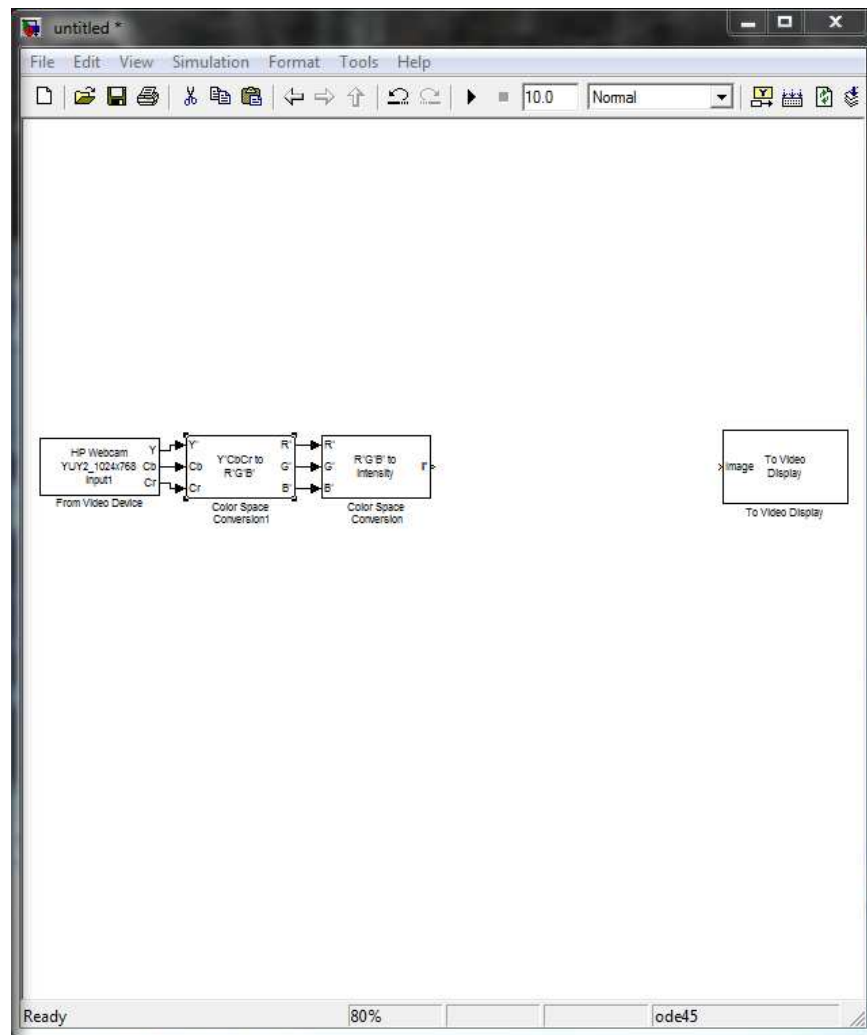


Figure 4

After the color conversion blocks, an edge detection block is needed. This is located in the analysis and enhancement library. Now there is a signal that is outputting all of the edges in the video. If the system was completed, the output would just show the edge detection. The goal is to overlay the original video and the edge detection, so the original R, G, B signal must be combined into one single signal. Three frame concatenation blocks must be attached to the R, G, B signal. Then the three frame concatenation blocks will be combined in a matrix concatenation block to achieve that single signal. This is all displayed in figure 5.

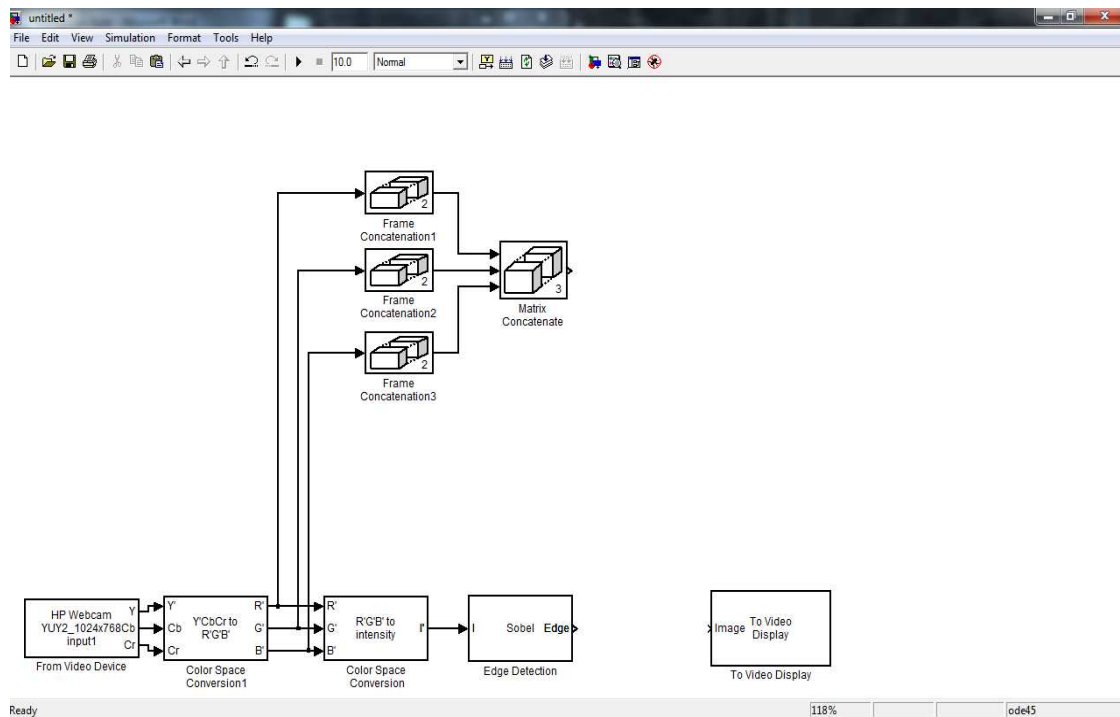


Figure 5

The final block needed in this design is a compositing block which will perform the overlay previously described. Once that is added, our system will be complete and the next phase is testing. Figure 6 shows our complete system.

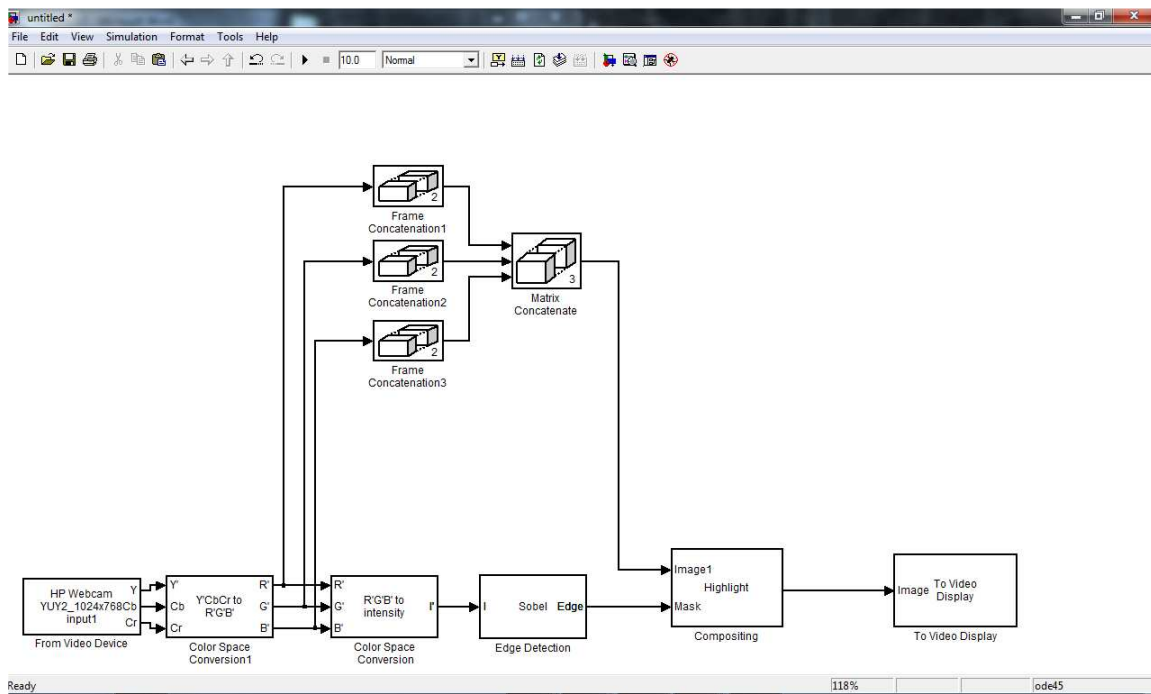


Figure 6

In the testing phase, not only is functionality tested but to understand the properties of all the blocks the parameters can be changed and the model can be retested. Figure 7 shows our algorithm function correctly.



Figure 7

Conclusions

This Simulink model displays edge detection on live video, but also shows the concepts of how to create the algorithm. Various edge detection methods can be tested, along with changing their parameters. Since it is easier to make changes and see the effects in this model, the future development of our system will improve because there will be less testing. Now an edge detection system for the Xilinx FPGA programming is more feasible knowing how an algorithm can be created.

The next step would be to create an edge detection system in Xilinx blocks. This is just the beginning of the project. Using our edge detection system teams will be able to create object detection, object classification, distance measuring, and more.

References

<http://www.mathworks.com/products/viprocessing/>

<http://www.xilinx.com/publications/archives/xcell/Xcell66.pdf>

http://www.mathworks.com/products/imaq/demos.html?file=/products/demos/shipping/imaq/demoimaqsl_edgedetection.html