Owen Shelton

CSCI 8810 Project 2

Spring 2011

Table of Contents

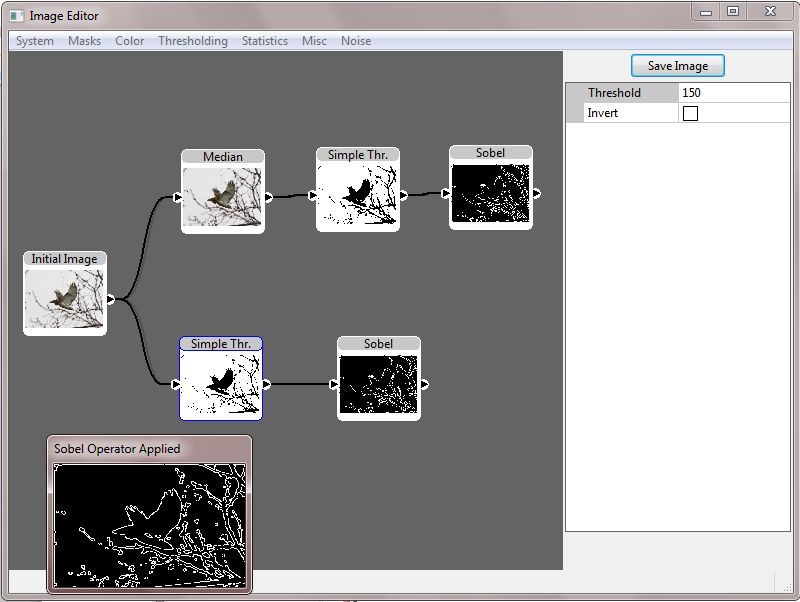
[Introduction 2](#_Toc289171375)

[Edge Detection 3](#_Toc289171376)

[Connected Component Labeling 7](#_Toc289171377)

[Pyramid 9](#_Toc289171378)

# Introduction

I came to the conclusion after the completion of project 1 that the use of a console program to perform experiments was both too awkward and time consuming to do any in depth experimentation. So for the purposes of this project I have implemented a simple GUI based on wxWidgets and a third-party open-source node connection control called connex. A basic screenshot from the UI can be seen below.

GUI Screenshot

Unfortunately, not all functionality from Project 1 has been re-implemented yet. Currently all operations that make use of or rely on histograms have yet to be implemented. It also currently only supports a single input image, and the rendering of the node connection control has a few minor bugs. However, implementing support for color images was trivial and all operators for which it would make sense to allow the use of color images support them. For this project the same image from project 1 has been used as the base image.

# Edge Detection

The edge detection operators all generate useful but very different results. Generally, the Sobel, Prewitt, and Kersh filters all generate very intense results and are much more sensitive to color change than either the Laplacian or Roberts filter. Once the pixel values in the Laplacian output have been normalized to 0-255 the output image is neither visually interesting or particularly helpful, but the underlying data used to generate the image is extremely useful.

|  |  |
| --- | --- |
| Original Image | |
| Figure 1, Roberts Operator Applied | Figure 2, Laplacian Operator Applied |
| Figure 3, Sobel Strength Operator Applied | Figure 4, Prewitt Operator Applied |
| Figure 5, Kersh Operator Applied | |

Figures 4 and 5 show very little visual difference between the Kersh and Prewitt operators, but there is a difference in the output between the two even if the human eye has a hard time picking it out. As can be seen by all of these results, using the results of the edge detection results directly may not be the best decision as there is still a lot of unnecessary information in the images that do not belong to any specific border and the edges are often quite indistinct (Figure 3 especially). Applying a thresholding operator after or even before can lead to clearer results.

|  |  |
| --- | --- |
| Figure 6, Sobel Operator -> Simple Thresholding T = 150 | Figure 7, Simple Thresholding T = 150 -> Sobel Operator |

As can be seen by Figures 6 and 7, thresholding greatly improves the distinctness and clarity of the edges, regardless of whether it is applied before or after the edge detection. Applying thresholding before the edge detection step can result in simpler and even more distinct results however as each pixel is calculated to either be an edge pixel or not an edge pixel; there are no potential edge pixels as there are when processing full color or grayscale images. These potential edge pixels cause the distortion and thicker edges seen in Figure 6. This data is not very valuable however as picking a good thresholding value for general use is extremely difficult, and typically thresholding is applied after the edge detection step as to maximize the amount of edge information available.

# Connected Component Labeling

Connected Component Labeling (CCL) is a very powerful tool and is extremely helpful at gathering basic information about possible regions of interest in an image. I was also required to implement this algorithm for CSCI 8820 (Computer Vision) and so my implementation includes some information not explicitly asked for in the instructions. My implementation uses a size threshold variable to weed out regions whose area are below a certain threshold and also calculates the centroid and the bounding box of each component.

|  |
| --- |
| Figure 8, CCL Applied with a size threshold of 0 and thresholded with T = 150. |
| Figure 9, CCL Applied with a size threshold of 500 and thresholded with T = 150. |

As can be seen in Figures 8 and 9, the size threshold can make a tremendous difference in the output and can be used to remove trivial regions or false regions that may correspond to patches of noise. It is not a replacement for filtering the image or choosing a good threshold value however as the size threshold cannot be used until the area of the region has been found which can lead to a large number of unnecessary calculations. The bounding box is useful for reducing the number of pixel comparisons needed when further processing is performed on a region and can help isolate a region from other regions.

It is important to note that the choosing of a good threshold value and/or thresholding technique will have a tremendous effect on the results of CCL. Even such a decision as to whether the objects of interest are going to be white or black after thresholding must be taken into consideration. My thresholding operators mitigate this problem by providing the ability to invert their results, but it is up to the user to decide whether or not it is necessary.

# Pyramid

The pyramid technique does provide some interesting applications for simplifying ROI detection, but the zero and first order hold schemes both present their own benefits and downsides. As always image resolution is a limiting factor and even when using bilinear interpolation there is only so much it can do when expanding a 16x16 image to 512x512. At that resolution there is simply not much information available.

|  |
| --- |
| Figure 10, node setup for Pyramid |

The Pyramid node has a special output port for the pyramid levels as well as ports for each individual level. The First and Zero order nodes also both have output ports for each individual level to facilitate further processing.

|  |  |
| --- | --- |
| Figure 11, Original input image padded to 512x512 (originally 512x385). | |
|  |  |
|  |  |
|  |  |
|  |  |
| Figure 12, First Order hold levels. | Figure 13, Zero Order hold levels. |

Hopefully the images in Figures 12 and 13 illustrate the differences between the First and Zero order hold techniques. It can be seen in both images that at 16x16 pixels, any kind of reconstruction is pretty much impossible.

It is extremely important to note that output of both the First and Zero order hold schemes are likely to need further processing before being useful. It may require many iterations of average filtering to mitigate the boxing effect in both techniques. Figures 14 and 15 show that median filtering can actually exaggerate the boxing effect. The exaggeration is even more pronounced when applied to images using the Zero order hold scheme.

|  |  |
| --- | --- |
| Figure 14, 3 Iterations of 9x9 Median Filtering on a 64x64 First Order Hold level. | Figure 15, 3 Iterations of 3x3 Median Filtering on a 64x64 First Order Hold level. |

The Pyramid technique can also offer significant performance improvements when used in conjunction with Connected Component Labeling (CCL). The CCL operation is computationally expensive and being able to find initial areas of interest that can be examined further in the full resolution and can avoid having to process the whole image.

My implementation has a limitation in the fact that the last column of data is lost when applying the First order hold technique. Whatever data is in the previous column is simply stretched into the last column. This could probably be avoided by offsetting the lower resolution data so that it resides mostly in the center of the image. This would require a complete rewrite sadly. Also, the bilinear filtering technique I use is not perfect and generates quite a few artifacts, most likely due to the fact that it is implemented slightly differently from normal due to going through the general steps of the First order hold technique.