Elliot Catalano
ecatala1@binghamton.edu

Method:

For both parts of the project, I attempted to implement a support vector machine (SVM) in conjunction with histograms of oriented gradients to differentiate the sample images into different classes of objects for object detection and pattern recognition.

In order to get the histograms of oriented gradients, here are the steps I implemented:

1. I changed the image to greyscale by multiplying the RGB values of the original image by a mask: red value by .114, green by .587 and blue by .299 values of each original pixel.

2. I implemented a gaussian blur filter to my image to reduce the noise in the image. I tried to minimize the effect of the blur to preserve as many edges as I could while using a Sobel operator.

3. I implemented horizontal and vertical Sobel operators to my blurred image, and created a combined sobel image with the equation D = sqrt(dx^2 + dy^2)

4. I then found the orientation of each pixel in my image by using the equation: orientation = atan(dy/dx)

5. For any given block, I then created a histogram of oriented gradients. This was done by first splitting my orientations into 9 blocks. For any orientation of value 0, 20, 40, 60, 80, 100, 120 and 160+, the magnitude (the value of the edges detected in the combined sobel image) was added to block 0,1,2,…8, respectively. For any value between these values, I would add half the magnitude to the blocks that they fell between.

Next…

For **part 1**, I then mined histogram data from different spots in my image and classified the histograms as either an abnormal cell or a normal cell. Normal cells also contained data for the white background.

The histograms were then collected and written to one of two files (one for abnormal, one for normal) and used to train an SVM.

In order to utilize an SVM in my project, I utilized the built in SVM library included in openCV.

In theory, the SVM would plot the histogram data from both categories into a 9 dimensional grid, and then plot an 8 dimensional hyperplane to split the data as efficiently as possible.

I used a linear kernel to plot the hyperplane onto my graph.

I then used this trained SVM to attempt to predict results.

I then iterated through my image based on a user-set block size, and highlighted any blocks detected as abnormal red, and any blocks detected as normal / background blue.

———————————————————-

For **part 2**, I utilized a similar method for collecting data, except this time, I split data into class1, class2, class3, and border. I then trained separate SVM's for each category.

An example of how I handled positive and negative data samples was, for the class 1 svm, I would read all class1 histograms as positive, and all class2, class3, and border histograms as negative.

I then looped through my image based on a user-set block size, and highlighted class 1 cells blue, class 2 cells green, class 3 cells red, and borders of cells purple.

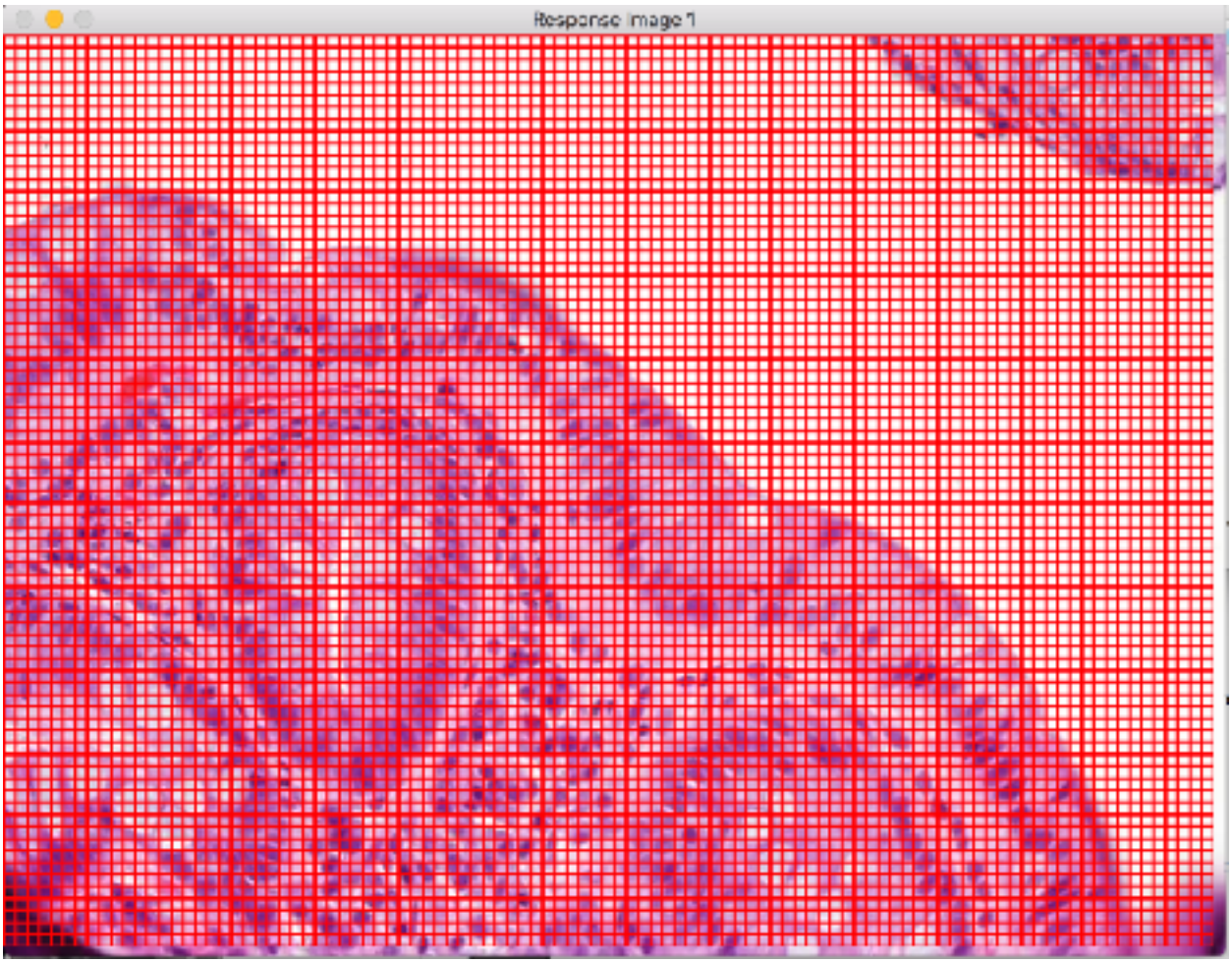
Results:

Unfortunately, testing multiple block sizes and multiple sized sets of data, I could not get the SVM's of either part to make proper predictions

What I believe was the issue was that the data sets between the positive and negative samples were too similar in some cases, so an adequate hyperplane may not have been created to differentiate between positive and negative predictions

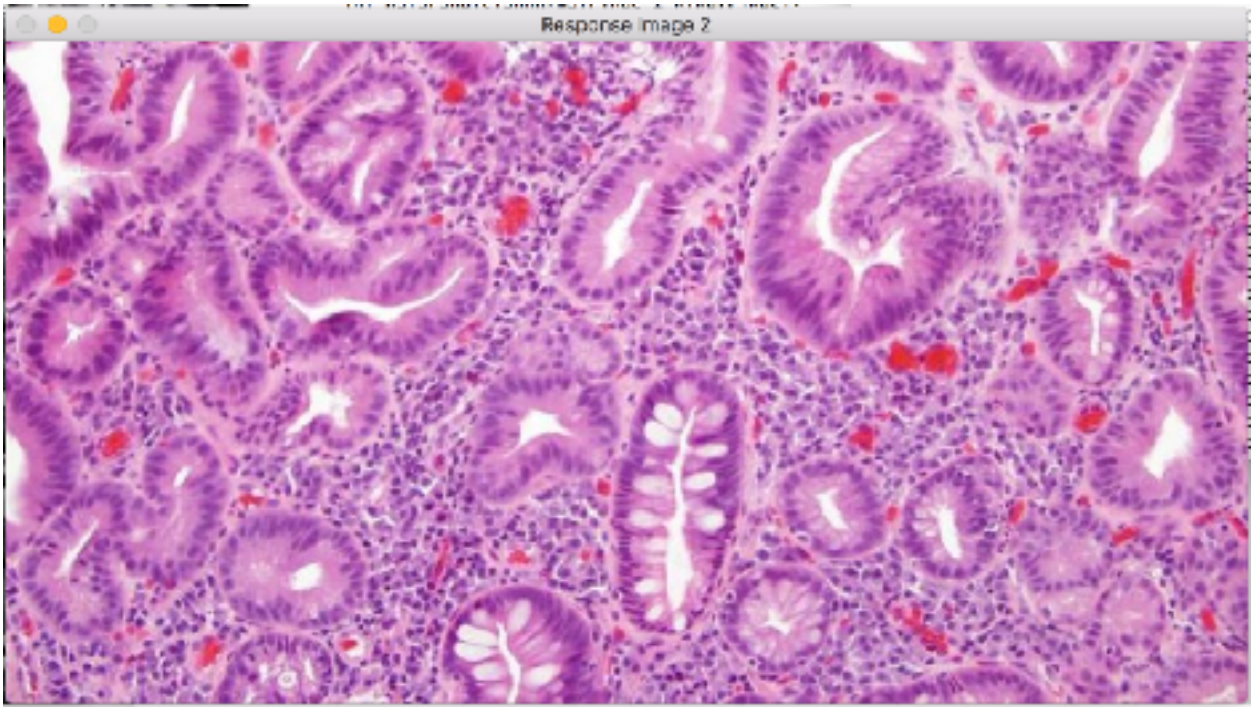
When making predictions, my SVM's yielded all -1 or all 1 (positive or negative predictions).

An example of the results for part 1:



As shown, all the blocks in the image were detected as abnormal cells.

Another example:



The SVM's in image 2 failed to detect any of the patterns.
In the terminal , the printout from this detection was a repetition of the line:
`Response 1: -1, Response 2: -1, Response 3: -1, Boundary Response: -1`

Overall, I thought this project was really interesting. My issue was that I spent most of my time learning about and trying to implement HoG detectors that I did not have enough time to properly test out my methods.

If I were to continue on my implementation of my project, I would attempt to utilize my SVM on some more descriptors mentioned as tips, such as variance, DCT components, histogram, grey scale texture, edges, or wavelet components.

`Sources used:`

http://www.codewithc.com/gaussian-filter-generation-in-c/

http://www.learnopencv.com/histogram-of-oriented-gradients/

http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients#Gradient_computation

https://www.quora.com/What-is-a-histogram-of-gradient-directions-in-computer-vision

https://en.wikipedia.org/wiki/Image_gradient

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html