# Mid-Term Project Report

For this mid-term project we were asked to identify and highlight, different bouldering routes, on a given sample, identified by the unique color of the holds on said route. To achieve this, I used a mixture of filtering and morphology to create highlighting luminosity rings around regions of holds of the same color. Then I would combine the highlight rings back onto the original image to represent the path chosen. In this report I’ll move step by step through each different method used, why I used it, and its resulting output. Overall, I was happy with the final result at the end I will list any places I feel I could’ve improved. For now, we can begin with the first step in the path highlighting process.

To begin my approach, I utilized the given demo code for color isolation to extract a given path based on user input of the color of said path fig.1. This extracted the general color regions using an open.cv tool of the desired “rocks” of the path. However this tool also would extract a noticeable amount of “bad regions” small spots that fell in the threshold but were not part of the actual desired path. Note that I opted out of using the green threshold entirely for generating almost more bad regions than good ones. A picture containing graphical user interface

Description automatically generatedFig.1

Next, I wanted to morph these regions, dilating them to be more useful for the highlighting process. First though it was necessary to remove the bad regions, to assure we were only focusing on the desired path. This was achieved utilizing the Open.cv opening morph tool to “clean up” the image. I used a 19 by 19 kernel to remove and really small outlier pixels while reserving the core of our desired path. Then I used the dilation morph tool with a 35 by 35 kernel to really expand those good regions fig.2

Fig.2

A screenshot of a computer

Description automatically generated with medium confidence

Next on the agenda was generating our gradient edges to be used as the highlighter outlines. For this I utilized the Sobel gradient edge filter on the luminosity component of our morphed image. This gave really nice usable outlines of the dilated regions fig3.

A picture containing text, screenshot

Description automatically generated Fig.3

The next part is the most computationally tedious but necessary step. The program would read through pixel by pixel the gradient edge output and look to achieve a few goals. First to produce a uniform resulting outline I converted all lumens greater than or equal to 128 flatly to 128; anything below was set to 0. Secondly to generate an “Average path line” I stored the greatest and lowest length values and stored the width locations of any pixels at those lengths that were greater than 128. Using linear equation principals and the averages of the top and bottom width locations I generated a slope and Y-intercept for an average path line. Because of the orientation of our (x, y) plane all length values correlate to negative y values. Then using the given width points as start an stop I graphed a line of pixels all to be set to 128 lumens to represent this average path line. Then I dilated the image to get the best outlining possible fig.4.A picture containing text

Description automatically generated

Fig.4

I chose 128 because its half of 255, I noticed in my testing Open.cv will rollover excessive lumen values, this lead to poorly defined boundary lines so 28 was a nice middle compromise that handled the rollover situation. Which takes us to our final step of converting the original image to the LUV space combining it with the modified edge extraction and displaying that result. Fig5

Diagram

Description automatically generated

This final result I was generally pleased with, the morph and filter trick gave pretty good results for all paths I chose to work with and wasn’t overly complex in execution. I will note with the Opening morph we sometimes lost small footholds on certain paths. As above the bottom most foot on yellow is completely missed. Another point of improvement could have been in drawing path lines between each different region instead of an overall path line. Possibly looking for y values that have more than some number of corresponding valid x values. Overall though the results were satisfying, and I enjoyed the project! The next page is the outputs of each color path I chose to look at minus Yellow displayed above [Orange, Blue, Purple, Pink] enjoy!

Diagram

Description automatically generatedDiagram

Description automatically generated

Orange Blue

Diagram

Description automatically generatedDiagram

Description automatically generated

Pink Purple