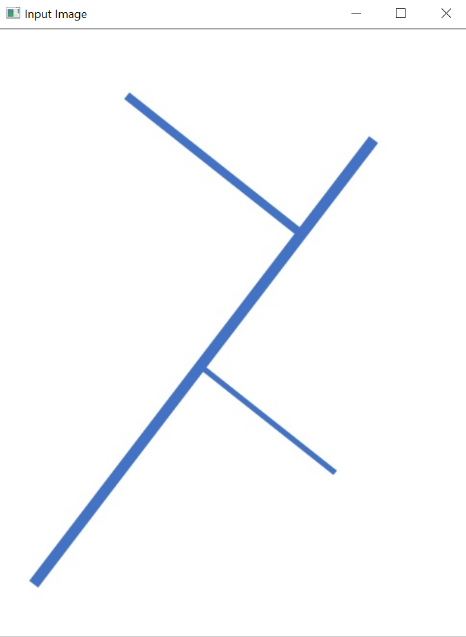
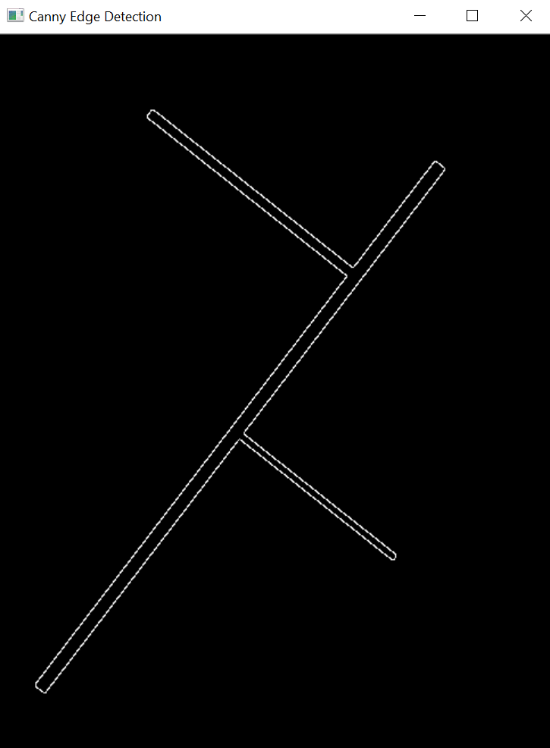
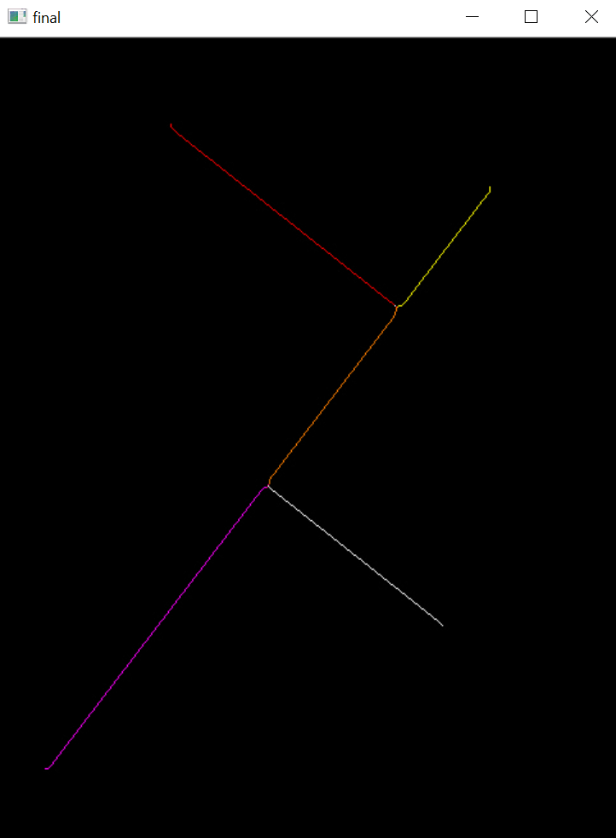
Cracking Analysis Paper

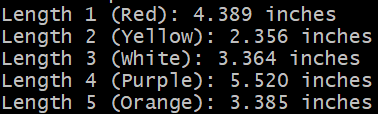
This program was written using the language python to take in an image of a material that has cracking in it and is able to determine statistics such as length and width of the lines. The program uses OpenCV, which is used to solve problems using image processing.

To start off, the user will type in the name of the file in which we will be analyzing the cracking. The user will also type in the how long and wide the image is along with the units. This will allow the program to properly calculate the results we are looking for. Once this is all done, the program will then be run using command line or an IDE that can run python programs. When the first image is inputted, the user will have the opportunity to either crop the image to the size they want or continue with the full image. This is particularly helpful when an image has extra stuff towards the borders that will mess up the calculations if included in the image. The image will be cropped using ROI which takes the coordinates of corners of the box that the user will drag across the image. Once the user is happy with the cropped image, the user will then press the esc button to continue.

The next step is to make the image binary. This means that every pixel in the image will be black or white. Our image will be white in the background and the cracking will be black. The code to go along with the development of the binary image is found under the function binary. Using this image, we will develop a new image that will get the edges of the cracking. By using canny edge detection, the resulting image will look like the image to the left. This will be helpful for the width calculations. The code that goes along with the development of the canny image is found in the function canny.

The next step is filtering the image. The program uses median filtering which is used to remove noise from the image. Noise, such as small dots in the background, will cause the program to not work and break. By removing the noise, the program can get accurate results for the calculations. With the image now filtered, the thinning function will be called. This function takes the filtered, binary image and returns a picture of the midline of the image. This skeletonized image will be used for finding the length of the cracking. When the thinned image is created, the complete function will be used to fill in any gaps. This is important because the length calculation only works if the skeleton line is continuous. This function leads to the bandw function, which is used to set up the skeleton image to be used to find the length. The bandw function will go through every pixel and make the image black and white.

The length function is then called. There are various parts to getting the length. The first step is the program loops through every pixel in the image and will get the red, green and blue values of the pixel. With these values, an if statement can be used to figure out if each pixel is part of the skeleton of the cracking. For each pixel that is part of the skeleton of the cracking, the getColor function is then called. This function takes in the x and y coordinates of the pixel and then will return a function that will include the pixels directly above, below, to the side or diagonal to the selected pixel, along with the number of pixels that fit those conditions. If a pixel, only has one pixel next to it, then it is an endpoint. This pixel will be colored blue. If a pixel has two pixels next to it, it falls in the middle of the line so it will be colored green. Lastly, if there are more than two pixels next to the selected one, it is categorized as a splitting point. These pixels will be colored blue.

The rest of the length function is used to get measurements of length for each segment of the cracking. Looping through the array of endpoints, if the pixel is part of the middle of the line, the length is updated. Once the pixels get to an endpoint, the total length is submitted. If the program reaches a splitting point, it is treated as an endpoint, and the splitting point is added as an endpoint. The length is outputted and then the program moves to the next endpoint. Once all of the lengths are measured, the output will look like the one to the left.

As for the width, the function uses all the slopes calculated from the three to five consecutive pixels in the skeleton. For each point, a width is calculated and then shown on the image. This is calculated by getting the slope of each point and then getting the perpendicular slope of each point. Then, the slope is made into a fraction. The x coordinate is added to both sides of the x direction and the y coordinate is added to both sides of the y direction. This is looped through until the width has hit the edge. This is shown in the images below.

