Assignment 1

Machine Learning for robotics (Elec0144)

Task 1

The Vision\_task1 file is made up of a Class object for the camera calibration. The images points and world points have been saved into separate csv files and I’m using the pandas library to extract them and use it in my python file. The result obtain for the intrinsic parameters aren’t really close to the expected result from the MATLAB code. The main reason I found while debugging my code was with the svd function that made the value of my H matrices quite small.

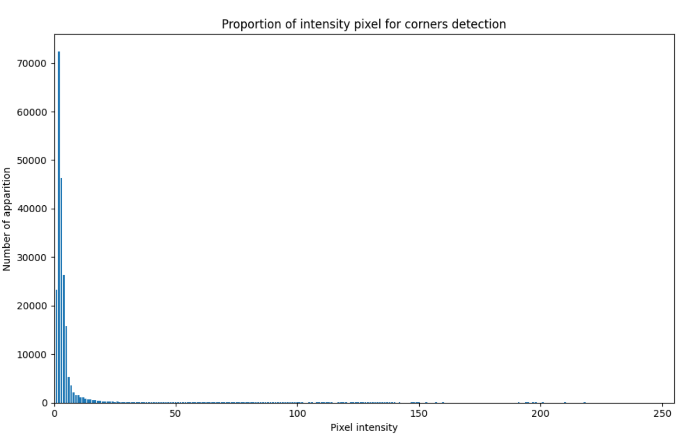
Task 3

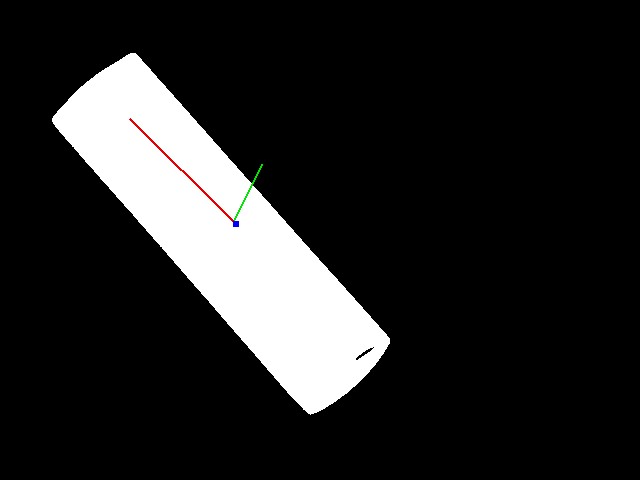
The Python file contains a Class object created to batch all the functions that would be required to perform pixel processing on the given object’s image. The program requires three main libraries to be launch properly, Image and ImageDraw from PIL to process the image, find its pixels array intensity and draw lines on it, the math library to access certain basics mathematical functions and numpy for array handling. As the pixel processing often requires pixel by pixel changes, three different instances of the Class P\_Processing are created to obtain distinct images for our results. The algorithm firstly finds the edges of the object by using the grey-scale version of it and by summing the Sobel x and y matrices. On the image they will appear in a grey-scale. The Sobel array is then used in a separate function where it will find the corners, the pixels value with the highest intensity. To find a coherent threshold, a dictionary of pixels’ intensity is created with the intensity as the keys and their number of apparitions as their values. It’s then sorted out and the key placed at of the dictionary becomes the threshold, to have a small amount of high pixels within this threshold. The grey scale array is then converted to an RGB array to show in red the corners of the object. This is the result from it:



Figure 2 : Corners detection shown in red with darker edges

Figure 1 : Edges and corners detection

We can see that the edges are shown in a grey scale and that corners are shown on the right with some red intensity value. The threshold was set up based on the number of repetition of an intensity value and created a gap of values for the threshold. Having only the corners was hard as the right-hand side of the object has a high value of intensity and could be detected as corners. Therefore this threshold has been kept to preserve most of the insight but will give as an answer 31 corners for this object.

Then, another instance of the P\_Processing Class was created to find the centre, the orientation, the major radius, and the minor radius (ellipse parameters) of the object. The first step of the related function was to obtain an array of intensity thresholds to have only black and white pixels. The pixels intensity have then been put in a dictionary storing the amount of time of appearance for a given intensity to then store a potential threshold for corners. To find the orientation of the object, the M00, M10 and M01 values were calculated, as well as the eigenvalues and eigenvectors of the threshold array. The angle found between the vector 1 and the horizontal vector was -138.55°. The angle between both vectors, vector 1 and vector 2, was also calculated which gave a 90° angle. The centre is obtained by calculating the Xc and Yc values, which gave us the point (233.23, 221.74). The radius *a* (in red) has been drawn from the centre by removing the length *a* to the centre coordinates. To obtain radius *b* a line has been drawn from the centre by removing to its coordinates the length *b* times the cosine of the angle between both vectors. The centre is shown in blue and has been drawn based on several pixels all located around the exact centre to see it more clearly. The result obtained from those computation is as follow:

The object is at an angle of -138.55° in terms of orientation from the horizontal line. Both radius has been drawn to better understand the position of the object.

Figure : Orientations and centre

Finally, to obtain the circularity of the object, a third class of P\_Processing was created. The perimeter was calculated with an array of black and white pixels and by looping over the pixels, it was determined by storing the position of the pixel where the one before or after has an opposite value (0 or 255). The perimeter therefore has fewer pixels drawn than for edge detection. As M00 gives us the area, the sum of all the pixels’ intensity, the sum of the intensity of the stored pixels gives the perimeter of the object.   
The circularity is then being calculated with the following equation: . In this case, the circularity is equal to 0.05, which shows that the object is quite elongated and doesn’t a circular shape. Here is the result of the perimeter calculation in terms of pixels representation:

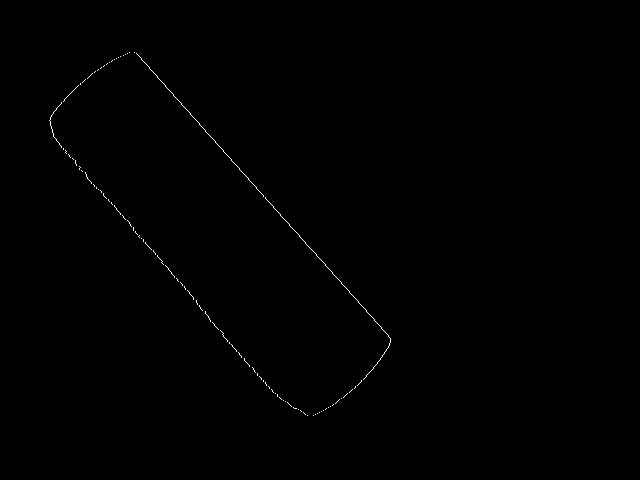


Figure 4: Perimeter of the object