CS472-Assignment1

Iason Tzimas - csdp1333

February 27, 2024

1

1.1

In order to derive a strategy of cropping the initial image into its subimages we will take a look at the mean intensity values per axis (x, y). Taking the first given image as a blueprint we get the following plots:

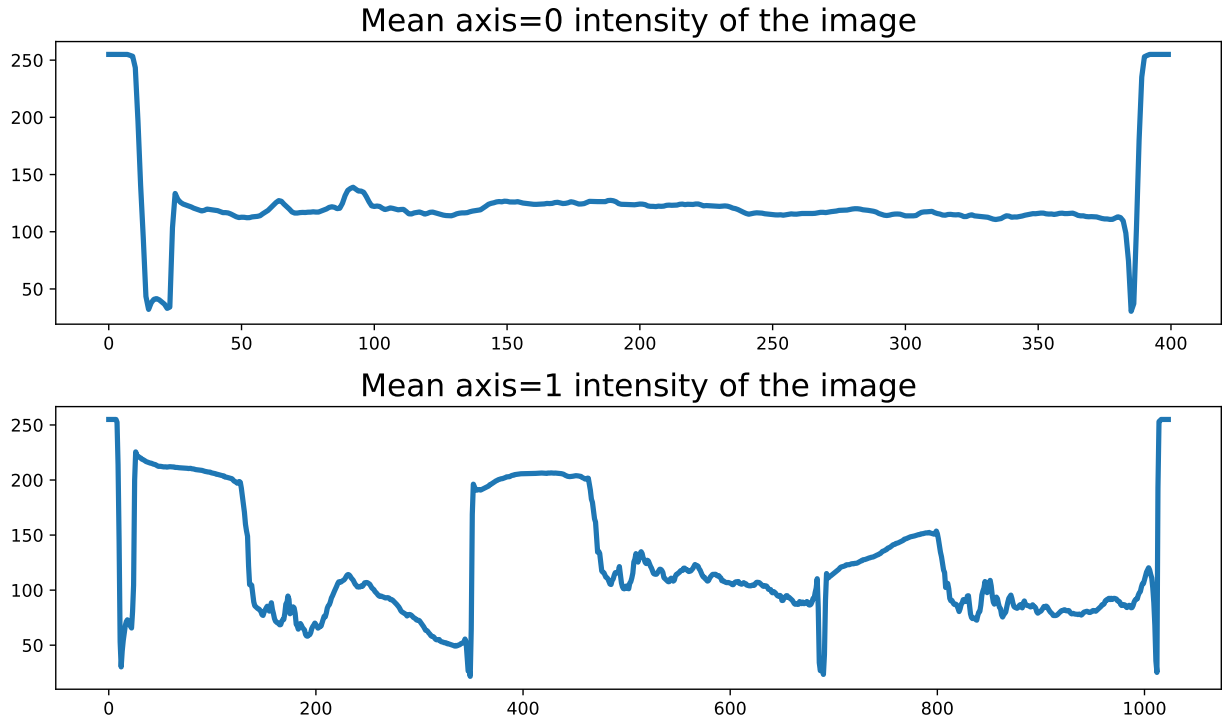

Figure 1: Mean intensity value per axis (x, y)

It is easily observed that the inner borders lying between the white outter border and the subimages is black. This leads to the selection of an easy to implement strategy, that of thresholding the Mean Intensity per axis vectors in order to the (x, y) coordinates where we enter the black border. The identified coordinates in both vectors are visualized as follows:

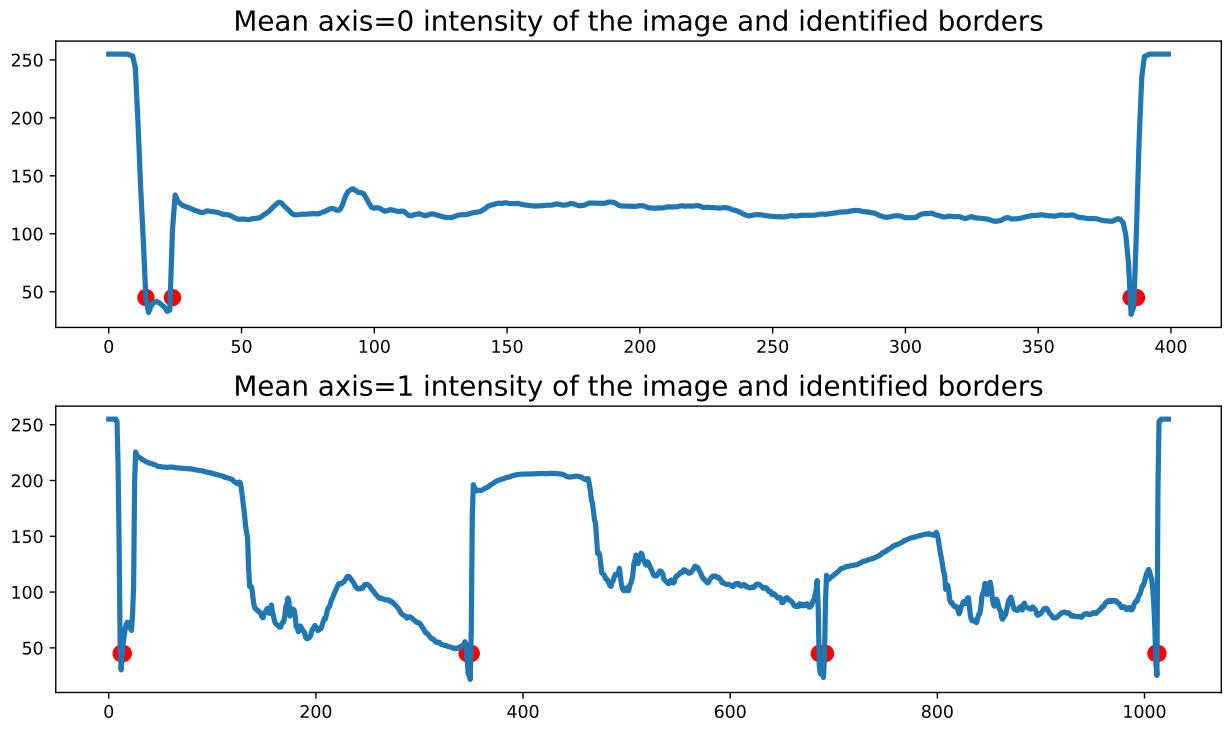

Figure 2: Mean intensity value per axis (x, y)

The coordinates with indices $(0, -1)$ (-1 refers to pythonic indexing) are removed because they refer to the points where we enter the black border coming from the white superborder and are thus of no interest.

By applying the above method to the blueprint image we get the following intial superimage and the resulting cropped subimages.

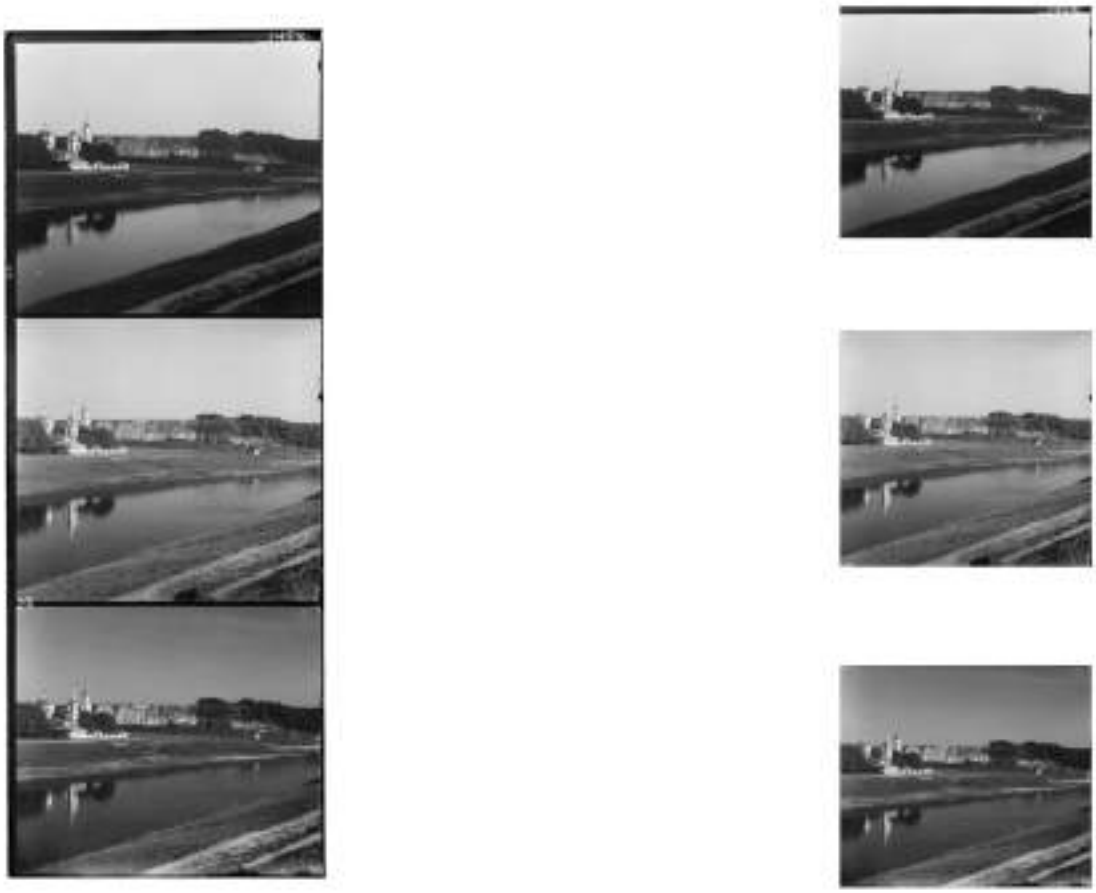

Figure 3: Original Image and subimages

1.2

We have now successfully separated the 3 color channels. Now, one channel is going to be padded with 30 pixels on each side and is going to be used as the main image. The remaining 2 channel Images are going to be used as templates and an exhaustive search of $60x60 = 3600$ iterations is going to be followed to identify the point (x_t, y_t) where the template best matches the image of reference. To account for the lack of scale invariance of the $\mathbb{L}^2$ norm Loss function, the Normalized-Cross-Correlation (NCC) or Pearson's Correlation Coefficient Metric is going to be utilized and is given by the following expression:

$$NCC = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k}(a[i,j]-\bar{a})(b[i,j]-\bar{b})}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{k}(a[i,j]-\bar{a})^2}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{k}(b[i,j]-\bar{b})^2}} \tag{1}$$

The values of the NCC metric for all iterations are as follows:

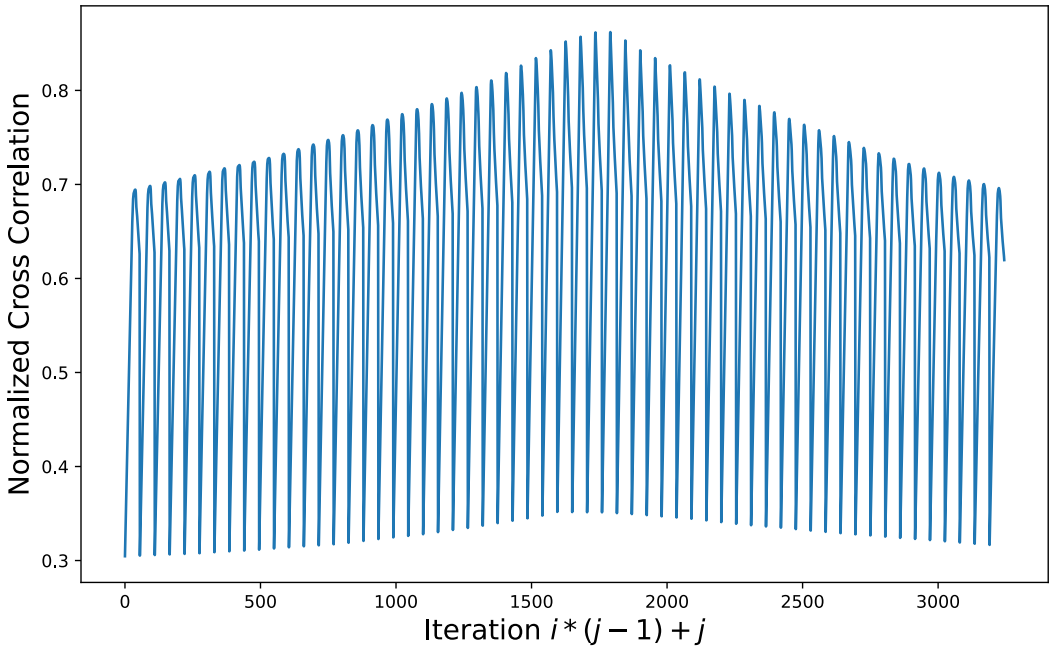

Figure 4: NCC value for each iteration

This obviously has a clear peak, which is selected to match the image of reference with the template channels.

Having identified the matching coordinates the 3-channels are stacked into a tensor to form the RGB image candidates. There is a total of 6 possible combinations. All these combinations together with the image considered to be the true RGB images are presented below. The matching coordinates $(x1, y1)$ and $(x2, y2)$ are included into the image captions

Figure 5: Aligned subimages in different sequences of channels. The alignment vectors are $(2, 1)$ and $(1, 6)$ with respect to the reference image

Figure 6: Aligned subimages in different sequences of channels. The alignment vectors are $(3, 5)$ and $(-8, 0)$ with respect to the reference image

Figure 7: Aligned subimages in different sequences of channels. The alignment vectors are $(1, 1)$ and $(-11, -6)$ with respect to the reference image

Figure 8: Aligned subimages in different sequences of channels. The alignment vectors are $(2, 3)$ and $(-8, -2)$ with respect to the reference image

Figure 9: Aligned subimages in different sequences of channels. The alignment vectors are $(0, 2)$ and $(-16, -9)$ with respect to the reference image

2

First, we acquire the "can.jpg" image:

Figure 10: Initial can image

2.1

For thresholding, we first acquire the Image intensity histogram and split it into two section first manually, and second following Otsu's method that is implemented from scratch. Otsu's method tries to maximize the intra-class variance.

The histogram is as follows:

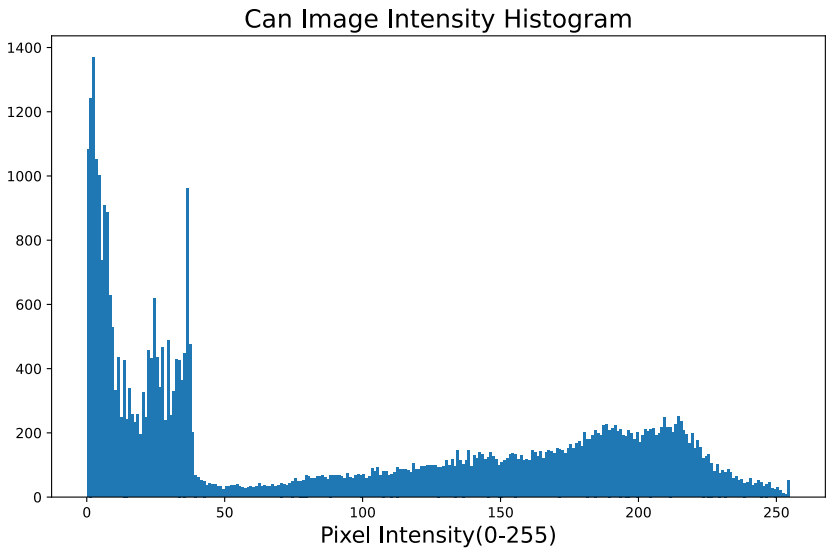

Figure 11: Image histogram

We can easily see that a threshold value of $\in (40, 50)$ separates well the two intensity distributions. A threshold of 50 is manually selected and results in the following:

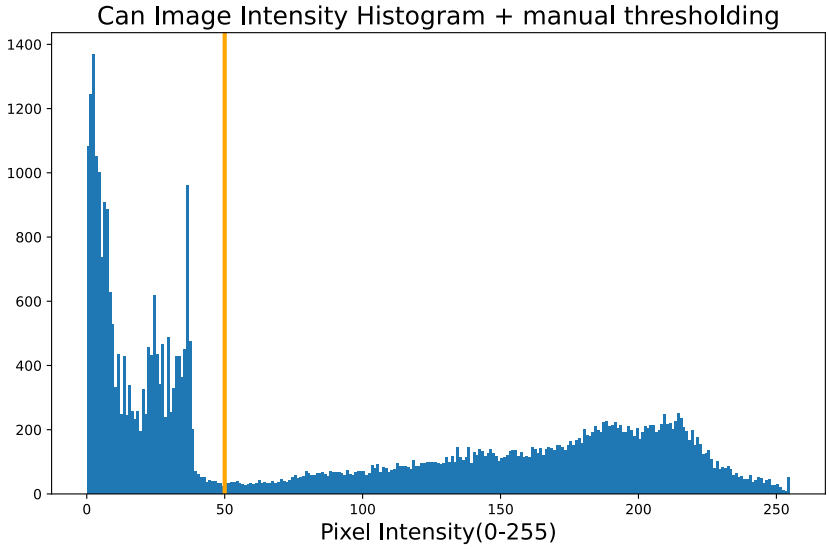

Figure 12: Image histogram and manual threshold

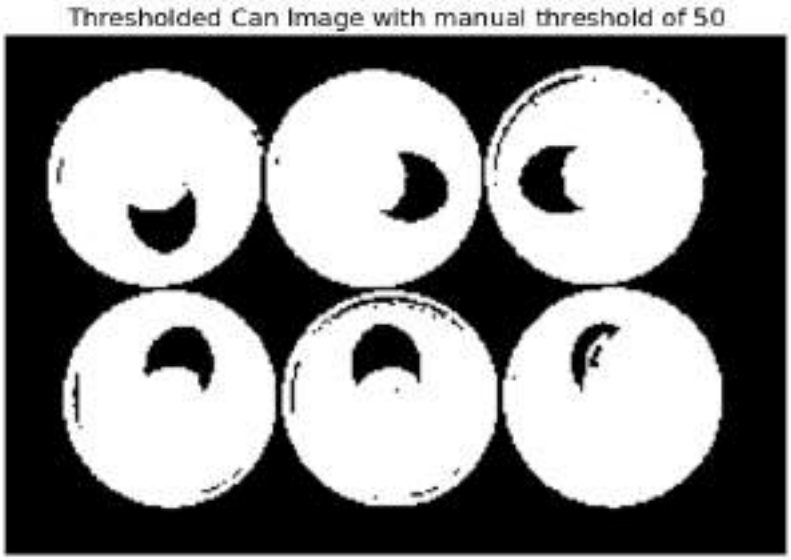

Figure 13: Manually thresholded image with a value of 50

Similarly, Otsu's method results in the following

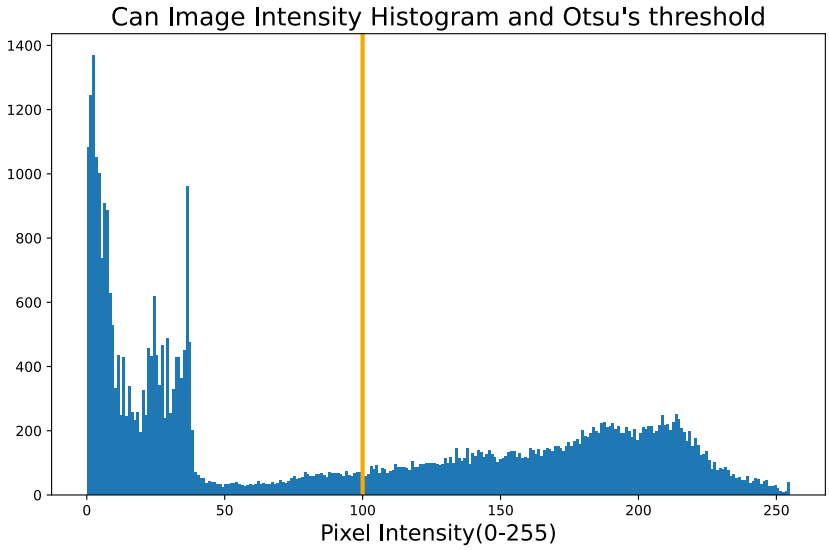

Figure 14: Image histogram and Otsu's threshold

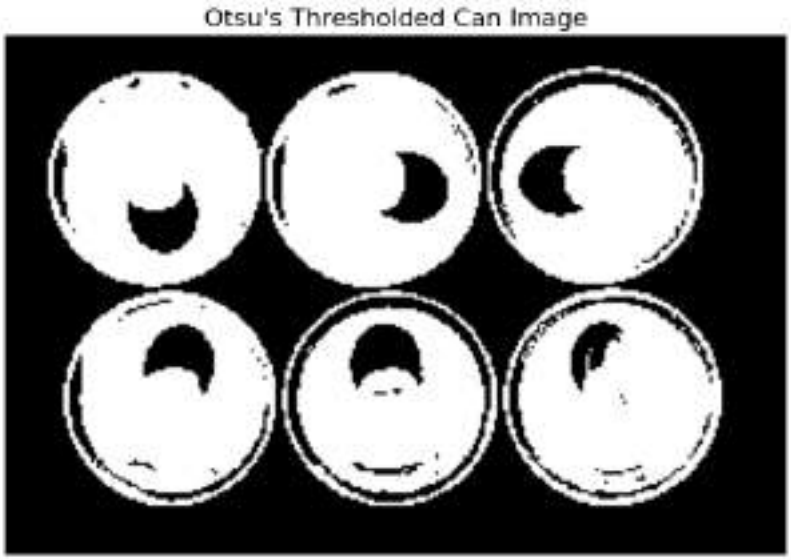

Figure 15: Otsu's thresholded image

We can see that Otsu's method introduces the circular disk within the can object segments which proved to be hard to remove using morphological filtering. Thus, the manually selected threshold of 50 is selected.

2.2

Next, morphological operations are going to be applied to the thresholded binary image to remove small holes (fill them in) and also remove extrusions. By a following a trial and error strategy for different sequences of closing, opening, erosion and dilation and with a selection of different kernel sizes, the best one is a sequence of $5-3$ kernel Opening, followed by a $3-3$ kernel closing, followed by a 3 kernel dilation, which leads to the following binary image.

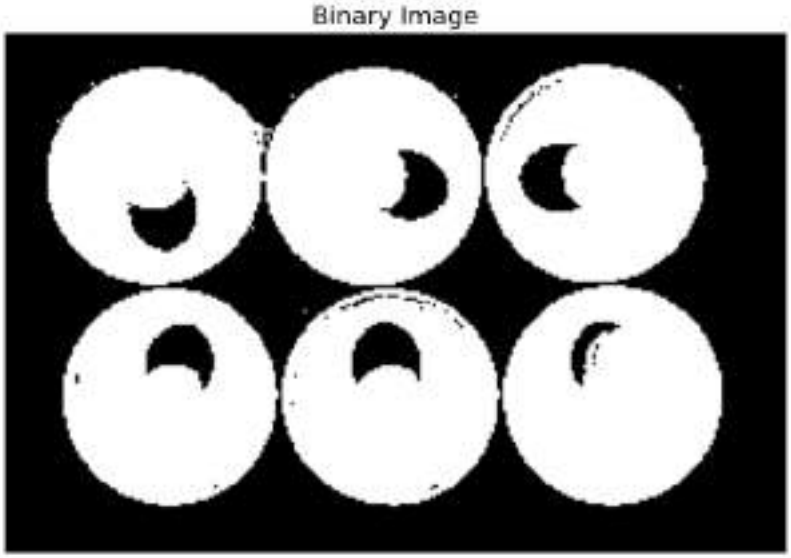

Figure 16: Binary Image

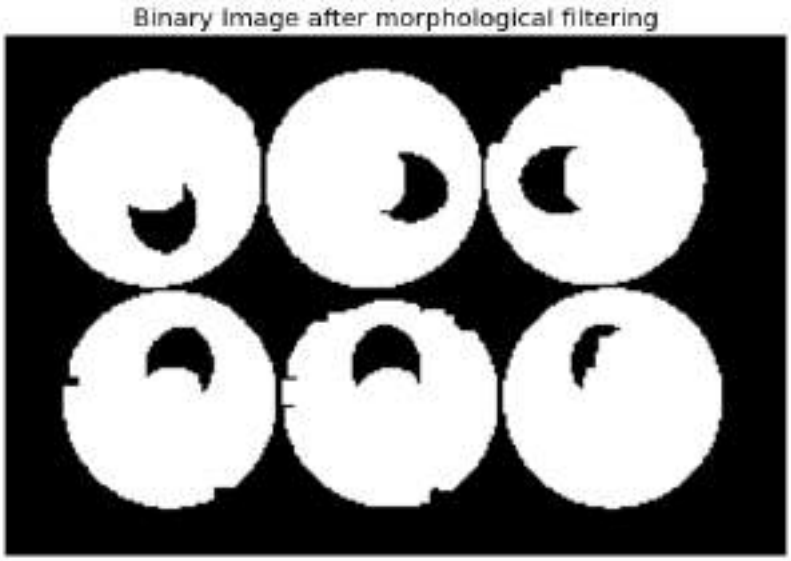

Figure 17: Binary Image after morphological filtering

One can now see that most holes and extrusions have been removed and the objects are well separated.

2.3

To identify different connected components/objects the 8-connectivity schema is going to be utilized, together with an implementation of the one-component-at-a-time algorithm from scratch. Essentially, this algorithms scans the entire binary image, pixel by pixel and when it encounters a foreground pixel that has not been labeled it performs Breadth-First Search by adding neighbor pixels that are foreground to the connected pixel queue and recursively searching the next pixels within the queue. Of course one has to use a label image array that stores the labeled vs non-labeled image pixels. The outcome of the algorithm is the following:

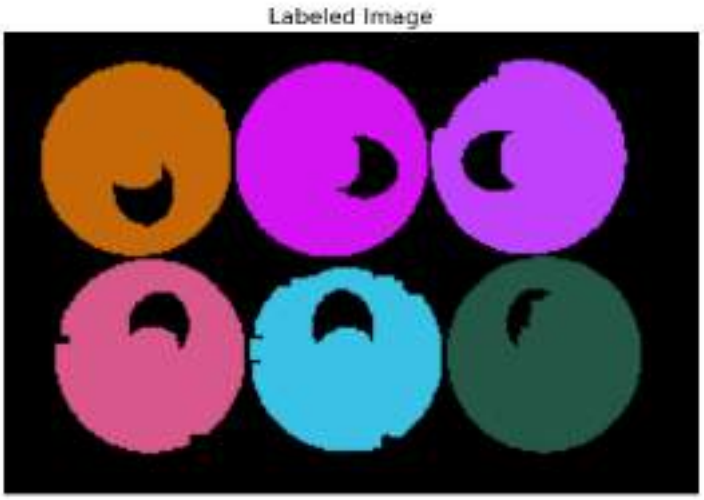

Figure 18: Labeled Image

The label image superimposed with the origianl is as follows:

Figure 19: Labeled Image

The algorithm does a great job in identifying the components.

2.4

The well known and studied in lectures formulas for moment extraction are used to estimate the centroids and rotations of the different objects within the picture and are the following:

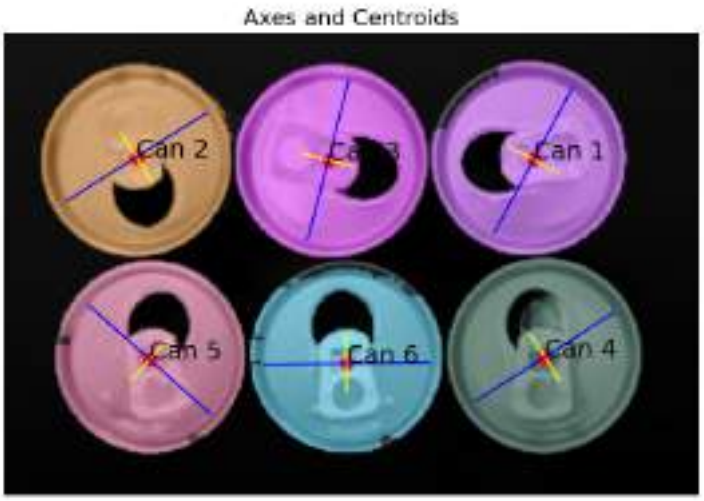

Figure 20: Labeled Image

Both major and minor axes are plotted. Most of the axes make sense and provide a rough approximation of the orientation of the cans, however, due to some artifacts introduced by the morphological filtering process, the axes are skewed in a non-trivial manner. This is due to insymmetries and the addition or removal of "mass" from ares that are further from the centroid such as the ones denoted in the following image:

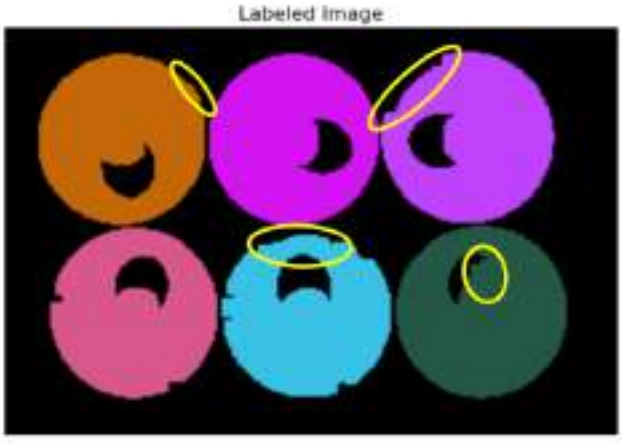

Figure 21: Labeled Image and notes

Finally, the Hue moments that are position, rotation and scale invariant are calculated using the formulas taught in the lectures. Roughly, due to these exact properties, Hue moments are a great, yet simple tool to vectorize a component/object with respect to its shape. Intuitively, similar shapes are going to be close or clustered in the $\mathbb{R}^7$ space spanned by the Hue moment scalars.

The visualized Hue moments of the can objects are as follows:

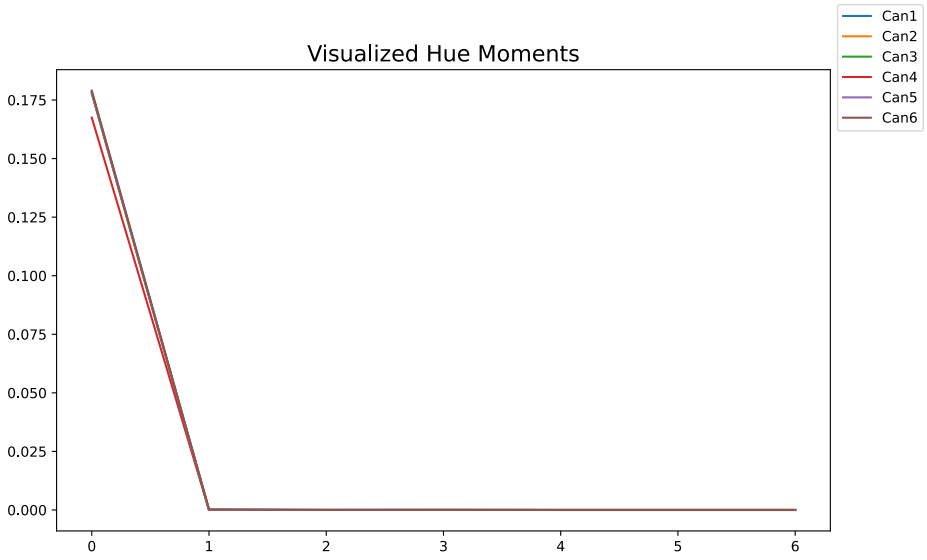

Figure 22: Hue moments of can objects

It is easily seen that all objects reside in the same neighborhood of the Hue moment space. The only object diverging slightly from the rest is object no4, which is indeed the one diverging most from the rest, due to the geometry of the half-opened hole that it has:

Figure 23: Can with different geometry

2.5

The same process is repeated for the manually acquired Image:

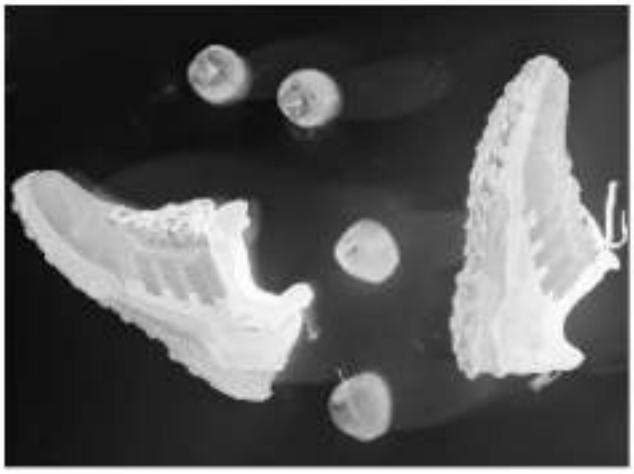

Figure 24: Manually acquired image

2.6

For thresholding, we first acquire the Image intensity histogram and split it into two section first manually, and second following Otsu's method that is implemented from scratch. Otsu's method tries to maximize the intra-class variance.

The histogram is as follows:

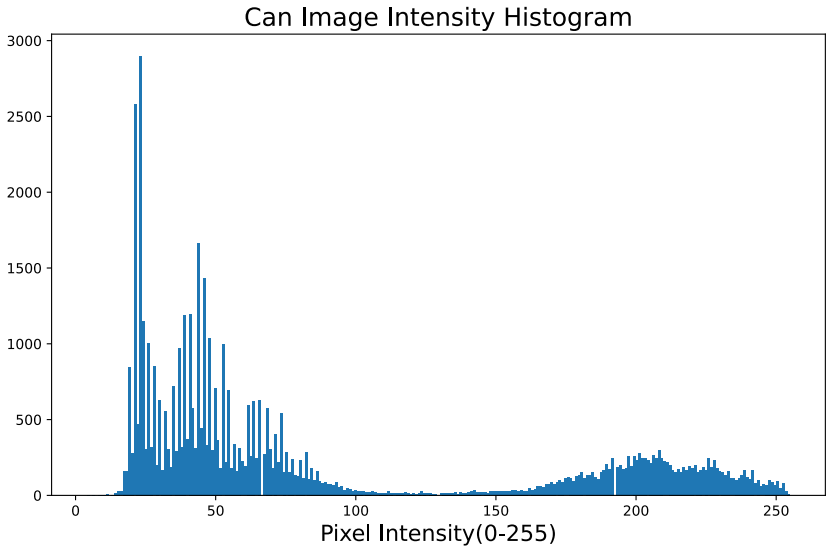

Figure 25: Image histogram

We can easily see that a threshold value of $\in (100, 150)$ separates well the two intensity distributions. A threshold of 125 is manually selected and results in the following:

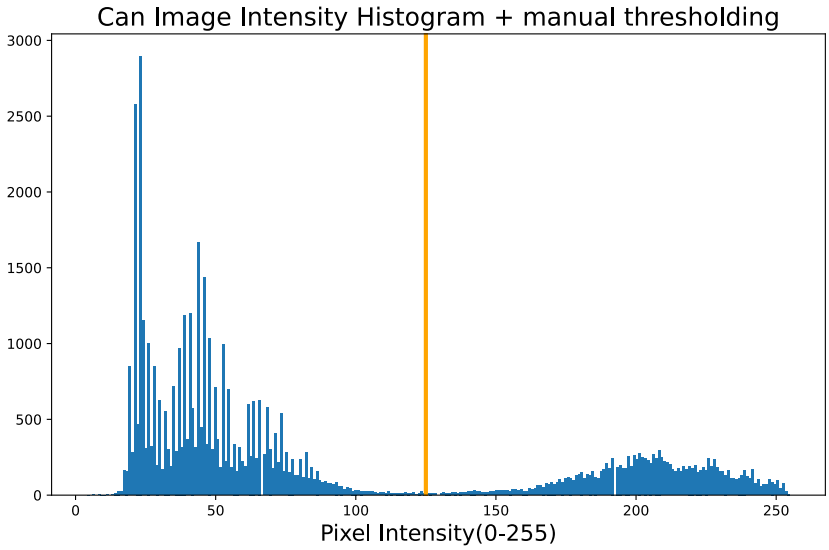

Figure 26: Image histogram and manual threshold

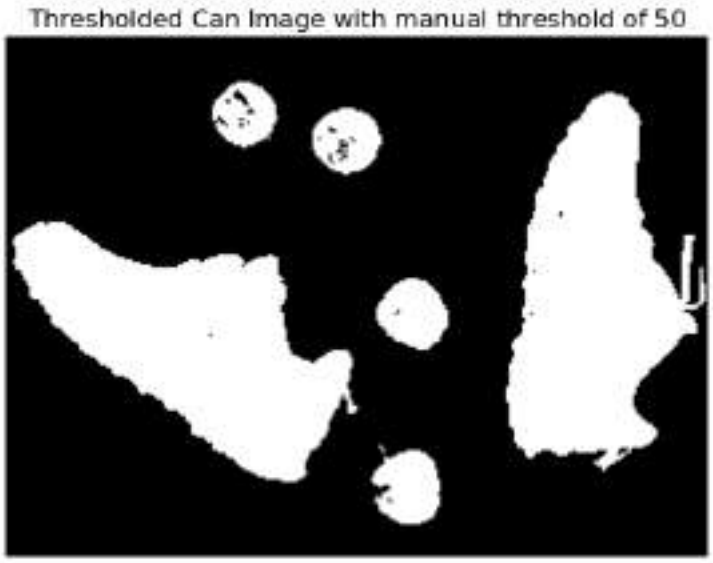

Figure 27: Manually thresholded image with a value of 50

Similarly, Otsu's method results in the following

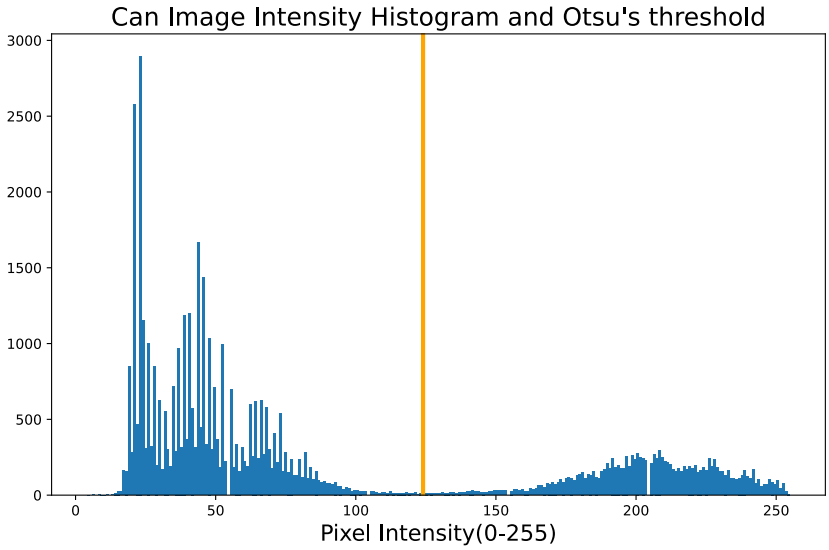

Figure 28: Image histogram and Otsu's threshold

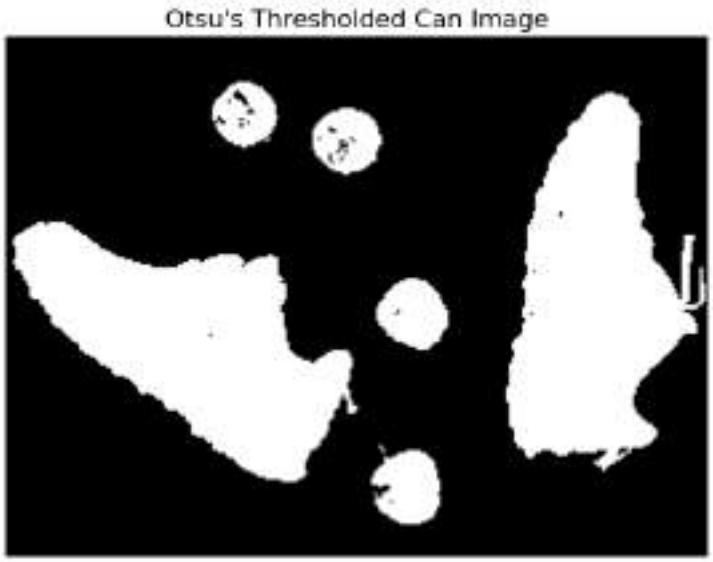

Figure 29: Otsu's thresholded image

Both methods introduce similar thresholds. Otsu's is selected to move on.

2.7

Next, morphological operations are going to be applied to the thresholded binary image to remove small holes (fill them in) and also remove extrusions. By a following a trial and error strategy for different sequences of closing, opening, erosion and dilation and with a selection of different kernel sizes, the best one is a sequence of $3-5$ kernel Opening, followed by a $3-3$ kernel closing, which leads to the following binary image.

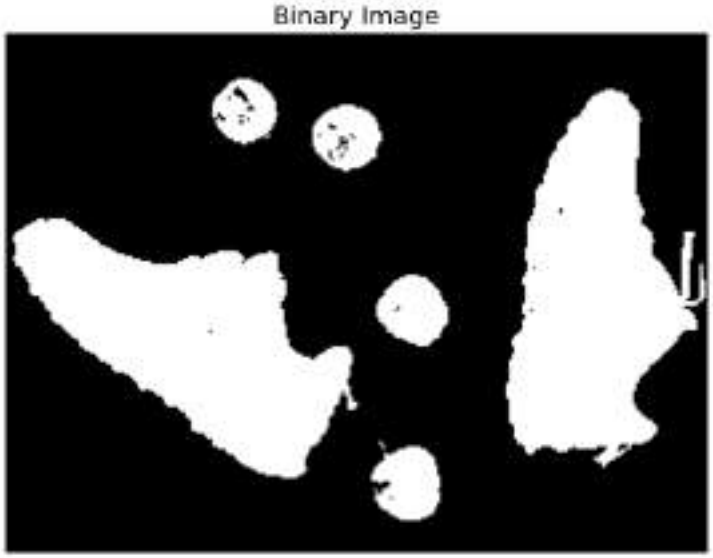

Figure 30: Binary Image

Figure 31: Binary Image after morphological filtering

One can now see that most holes and extrusions have been removed and the objects are well separated.

2.8

The same connected segments identification algorithm described above is used, leading to the following labeled image.

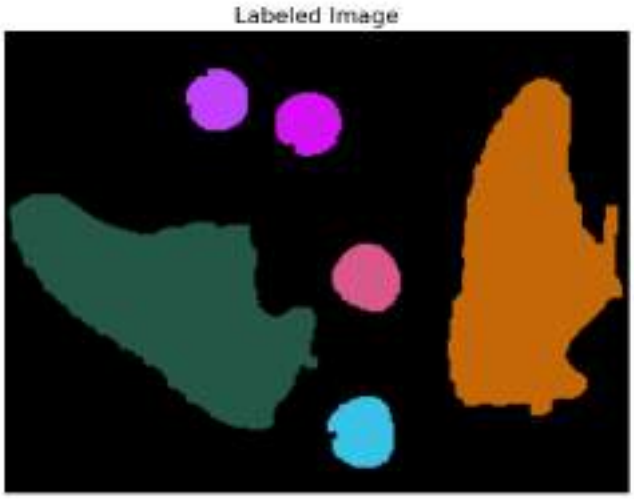

Figure 32: Labeled Image

The label image superimposed with the original is as follows:

Figure 33: Labeled Image

The algorithm does a great job in identifying the components.

2.9

The well known and studied in lectures formulas for moment extraction are used to estimate the centroids and rotations of the different objects within the picture and are the following:

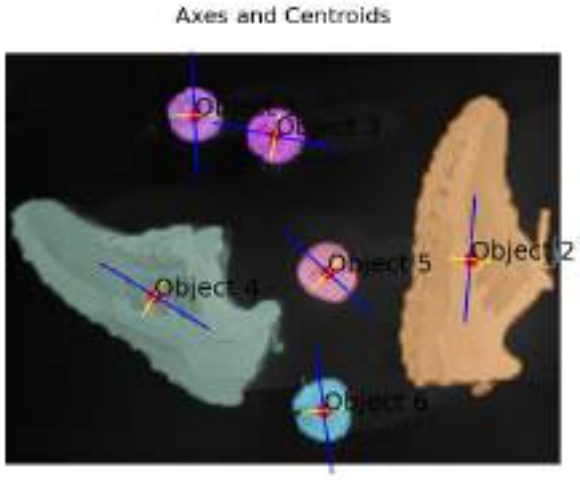

Figure 34: Labeled Image

Both major and minor axes are plotted. Most of the axes make sense, with perhaps the exception of apples which are roundish and symmetric leading to an ill defined set of major-minor axes.

Finally, the Hue moments that are position, rotation and scale invariant are calculated using the formulas taught in the lectures. Roughly, due to these exact properties, Hue moments are a great, yet simple tool to vectorize a component/object with respect to its shape. Intuitively, similar shapes are going to be close or clustered in the $\mathbb{R}^7$ space spanned by the Hue moment scalars.

The visualized Hue moments of the can objects are as follows:

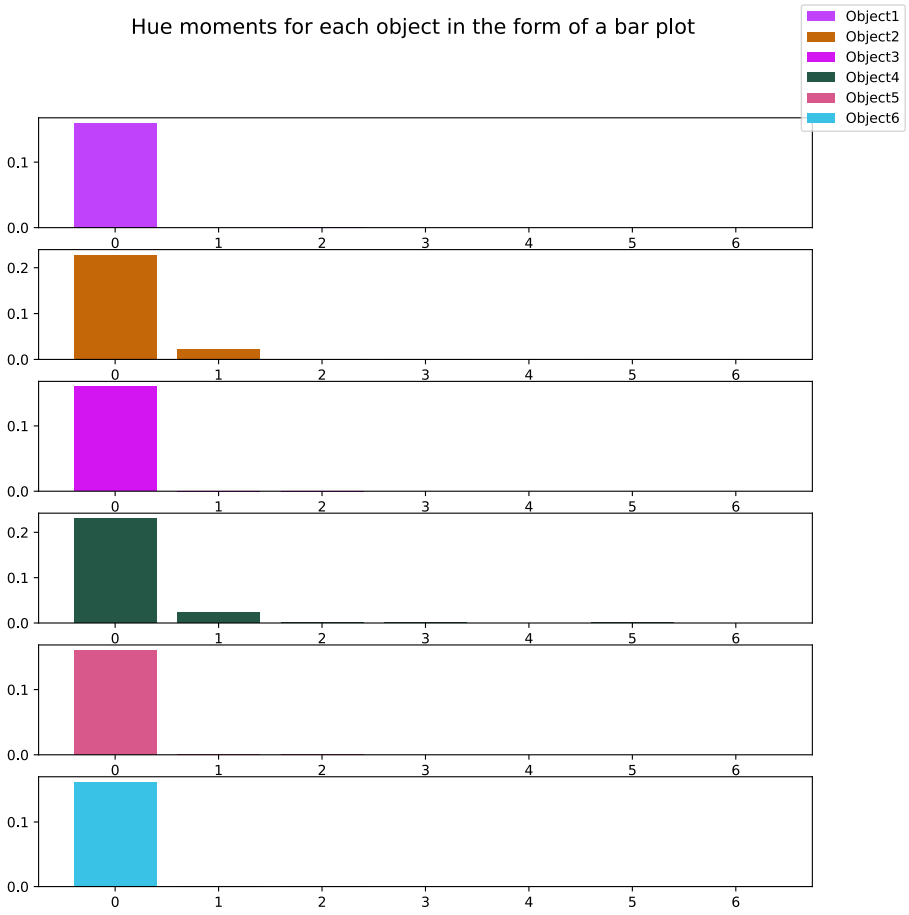

Figure 35: Hue moments of can objects

It is easily seen that Objects 2, 4 that refer to the two shoes have a different Hue moment vector and are identical to one another. The same holds for apples. It is obvious that two separate clusters are introduced and one can easily implement an algorithm to identify where each object belongs