ENPM 673: Perception for Autonomous Robots - MidTerm Exam

Aswath Muthuselvam
aswath@umd.edu
118286204

16th March 2022

1 Question 1: Counting coins

1.1 Separate the coins

The coins are separated by lines of varying thicknesses, hence it can be removed by using a structuring element filled with zeros. I separated the coins using erosion. I used a kernel of size 20x20 with one iteration of erosion.

1.2 Counting the coins

Now that the coins are separated, I used opencv's find-contours() function to find all the 24 convex shapes. The findcontours() function takes all non zero pixels as 1 and zero pixels as zero, hence, I converted the eroded image to grayscale and thresholded it to remove any darker pixels. the The number of detected contours detected is also the number of coins present in the original image. The number of coins detected will be printed by the question1.py script that is included with this report. Figure 1 shows the complete process of erosion and contour detection.

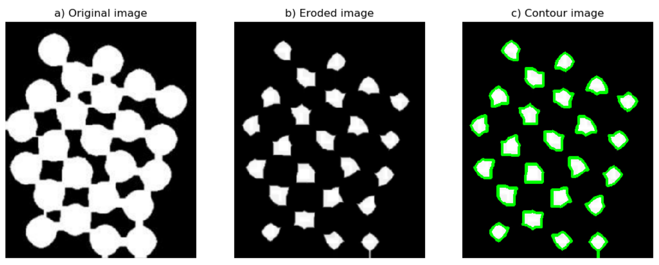

Figure 1: Counting coins in the image

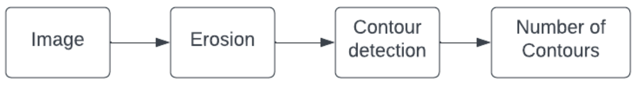

Figure 2: Block diagram of coin counting pipeline

2 Question 2: Image Stitching

There are two images given, one of the images is taken by translating the camera to the left side. We are going to stitch both of them together.

First, I converted the images to grayscale. Then, I used Feature detection to find significant Features in the image. Using the keypoints and descriptors obtained, I matched the keypoints using a Brute force matcher with L1 norm, this gives the relation between the points in the Image A and Image B. I used opencv findHomography() to find the Homography between the source and destination points. Then, with the Homograhy matrix, I warped the images using warpPerspective() function. Figure 3 shows the complete process of Feature detection, Keypoint Matching, finding Homography and Warping the final image.

The homography transformation matrix is given by:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

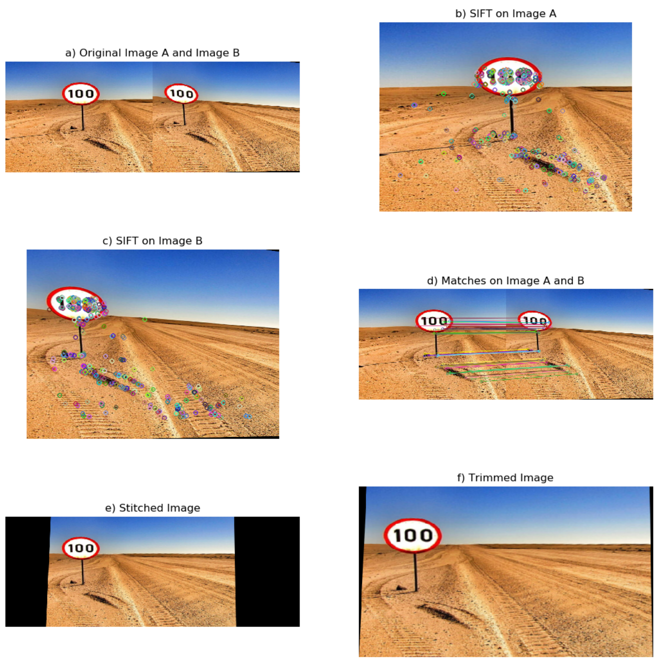

Figure 3: Stitching two images

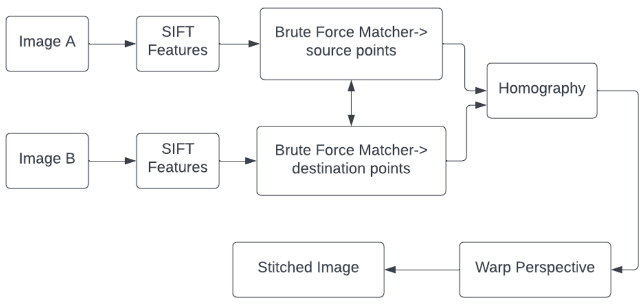

Figure 4: Block diagram of Image stitching pipeline

3 Question 3: Camera parameters

1. What is the minimum number matching points to solve this mathematically?

Answer: 6 points are required to solve this.

2. What is the pipeline or the block diagram that needs to be done in order to calibrate this camera given the image above.

Answer: The block diagram is shown below in Figure 5.

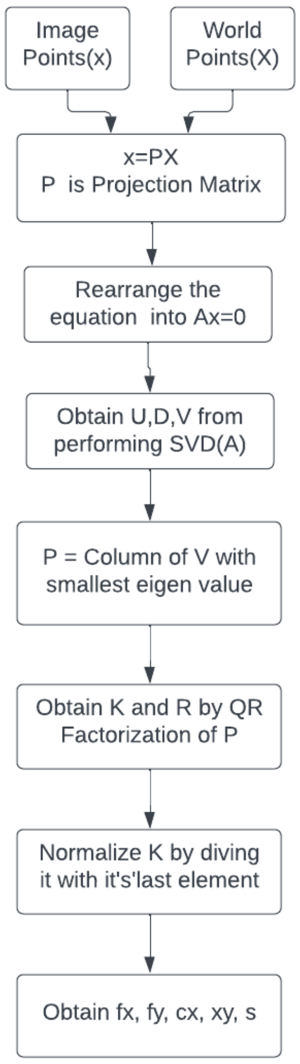
Image Points(x)
World Points(X)
x=PX
P is Projection Matrix
Rearrange the equation into Ax=0
Obtain U,D,V from performing SVD(A)
P = Column of V with smallest eigen value
Obtain K and R by QR Factorization of P
Normalize K by diving it with it's'last element
Obtain fx, fy, cx, xy, s

Figure 5: Block diagram of Finding Camera matrix - K

3. First write down the mathematical formation for your answer including steps that needs to be done to find the intrinsic matrix K

Answer: x is the coordinates of points in the image. X is the coordinates of points in the world frame. P is the Projection matrix.

$$x = PX$$

P can be decomposed as

$$w \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

By the pinhole camera model, P reduces to:

$$\mathbf{xi} \times \mathbf{PXi} = \begin{bmatrix} v^{\mathrm{i}}_{\mathbf{i}}\mathbf{p}3^{\mathrm{T}}\mathbf{Xi} - w'_{\mathbf{i}}\mathbf{p}2^{\mathrm{T}}\mathbf{Xi} \\ w'_{\mathrm{i}}\mathbf{p}1^{\mathrm{T}}\mathbf{Xi} - u'_{\mathbf{i}}\mathbf{p}3^{\mathrm{T}}\mathbf{Xi} \\ u'_{\mathrm{i}}\mathbf{p}2^{\mathrm{T}}\mathbf{Xi} - v'_{\mathbf{i}}\mathbf{p}1^{\mathrm{T}}\mathbf{Xi} \end{bmatrix}$$

The above equation can be rewritten as:

$$\begin{bmatrix} \mathbf{0}4^{\mathrm{T}} & -w\mathrm{i}'\mathbf{X}\mathrm{i}^{\mathrm{T}} & v\mathrm{i}'\mathbf{X}_{\mathrm{i}}^{\mathrm{T}} \\ w'_{\mathbf{i}}\mathbf{X}\mathrm{i}^{\mathrm{T}} & 04^{\mathrm{T}} & -u'_{\mathrm{i}}\mathbf{X}_{\mathrm{i}}^{\mathrm{T}} \\ -v'_{\mathbf{i}}\mathbf{X}\mathrm{i}^{\mathrm{T}} & u\mathrm{i}'\mathbf{Xi}^{\mathrm{T}} & \mathbf{0T}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = 0$$

This can be represented in the form: $Ax = 0$

We then perform SVD of A. The last column of V is the P matrix. V can be reshaped into 3x4 matrix. In a homogeneous equation such as $Ax = 0$, any vector x in the null space of A is a solution. Hence any column of V whose corresponding singular value is zero is a solution. In our case, since the data is from real world with noise added to it, we can take the column with the least Eigen value.

The K matrix is of the form:

$$K = \begin{bmatrix} f & 0 & p_x & 0 \\ f & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The matrix obtained is:

$$K = \begin{bmatrix} 34.1030 & 5.8943 & 755.1036 \\ 0 & 34.4705 & 8.6312 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

4 Question 4: K-Means segmentation

Since the classification is based on the color, the feature space for the K-means is the color space, i.e. Red, Green and Blue components. This means, that, the distance between the feature points is calculated between the components of the 3 color spaces. I did think about converting the image to LAB and segmenting in that feature space. However, RGB works fine as well. Before clustering, I resized the image to 100x100 pixels before processing it. Figure 6 shows the colors present in the traffic signal image and Figure 7 shows the plot of the 4 Means that the algorithm converged into.

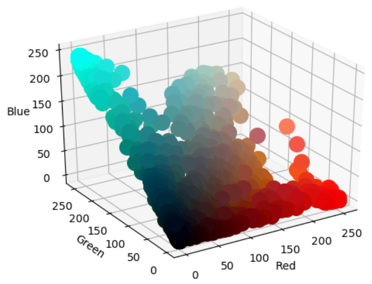

Figure 6: Color space of the input image

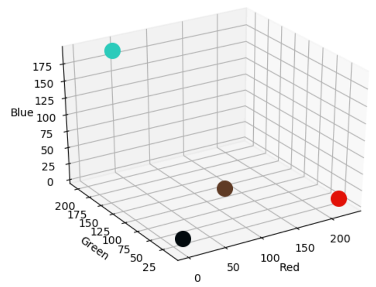

Figure 7: K-Means, K=4

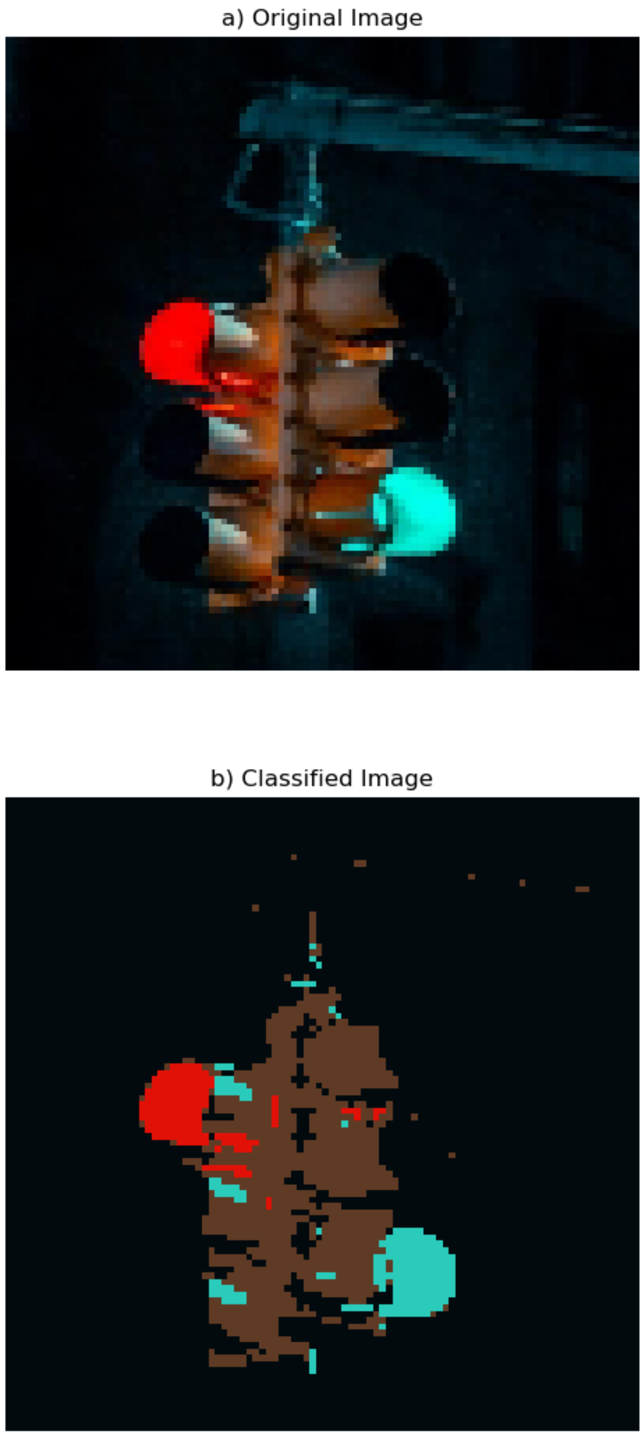

Figure 8: K-Means segmentation, K=4

The algorithm of K-Means that I implemented is described as follows. We are given a set of feature points X, which we want to cluster it into K classes. We start by creating a Dictionary D with K keys, each key will contain a list of values which will be obtained from the feature points based on the method which will be described next. We also have a list of Means M of size K containing the means of each cluster. Initially, the values of each means m_i are assigned randomly. Next, we loop through each feature point and computes its distance form each of the means, the nearest distance d is assigned to the respective class in the dictionary. The distance between a given feature point and mean can be calculated with Euclidean or Manhattan distance. The mean of each classes in the dictionary is computed and updated to the respective mean m_i. The above described procedure can be iterated for any number of time steps T to obtain a closer fit to the data, or it can be stopped when the means m_i are not changing with every iteration.

The output is shown of 4 class - classification is showed in Figure 8. The 4 classes are black, red, blue and brown color, image(b) shows the classes in a discretized manner.