MidTerm

PERCEPTION FOR AUTONOMOUS ROBOTS

March 16, 2022

Instructors:

Dr. Samer Charifa

Student:

Akash Ravindra

Semester:

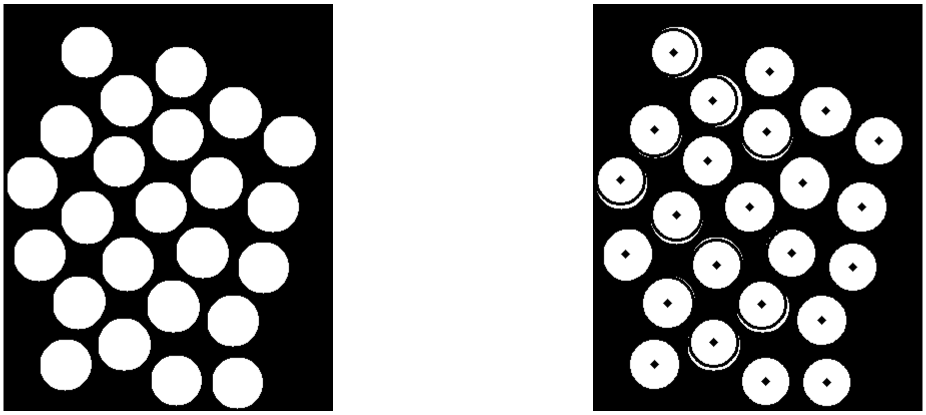
Spring 2022

Course code:

ENPM673

Contents

List of Figures



((a)) ((b))

Figure 1: **Question 1:** 1(a) shows processed image with all the coins in the image separated. 1(b) shows center of each circle on the processed image

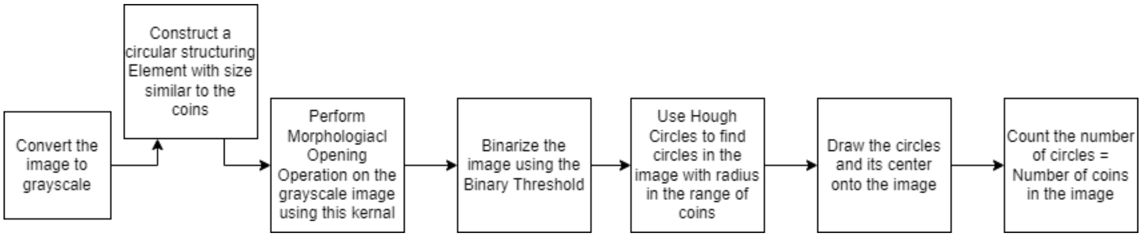

Convert the image to grayscale
Construct a circular structuring Element with size similar to the coins
Perform Morphologiacl Opening Operation on the grayscale image using this kernal
Binarize the image using the Binary Threshold
Use Hough Circles to find circles in the image with radius in the range of coins
Draw the circles and its center onto the image
Count the number of circles = Number of coins in the image

Figure 2: Pipeline for Question 1

1 Q1 X-ray of old coins.

1.1 Separate each coin from each other

This can be achieved by a series of morphological operations. The opening operation in specific along with a circular kernel can be used to remove the noise that exists between the image. Refer image 1(a) for results

```
kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE,(50,50))
closed = cv.morphologyEx(image, cv.MORPH_OPEN, kernel)
```

1.2 Count the number of coins in the image

Once the image has been cleaned up using morphology, Hough Transforms, specifically Hough Circles can be used to detect, count and draw circles on all of the possible coins in the image. The

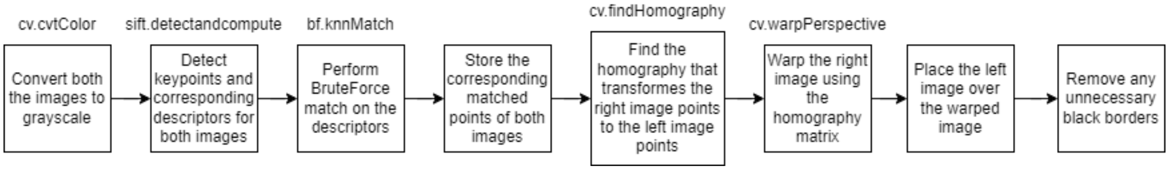

cv.cvtColor
Convert both the images to grayscale
sift.detectandcompute
Detect keypoints and corresponding descriptors for both images
bf.knnMatch
Perform BruteForce match on the descriptors
Store the corresponding matched points of both images
cv.findHomography
Find the homography that transformes the right image points to the left image points
cv.warpPerspective
Warp the right image using the homography matrix
Place the left image over the warped image
Remove any unnecessary black borders

Figure 3: Pipeline for Question 2

pipeline is described in the figure 2

```
circles = cv.HoughCircles(closed, cv.HOUGH_GRADIENT, 1,20, param1 = 70,
                          param2 = 15, minRadius = 22,maxRadius = 50)
```

Refer image 1 for results. A total of 24 coins were found in the image.

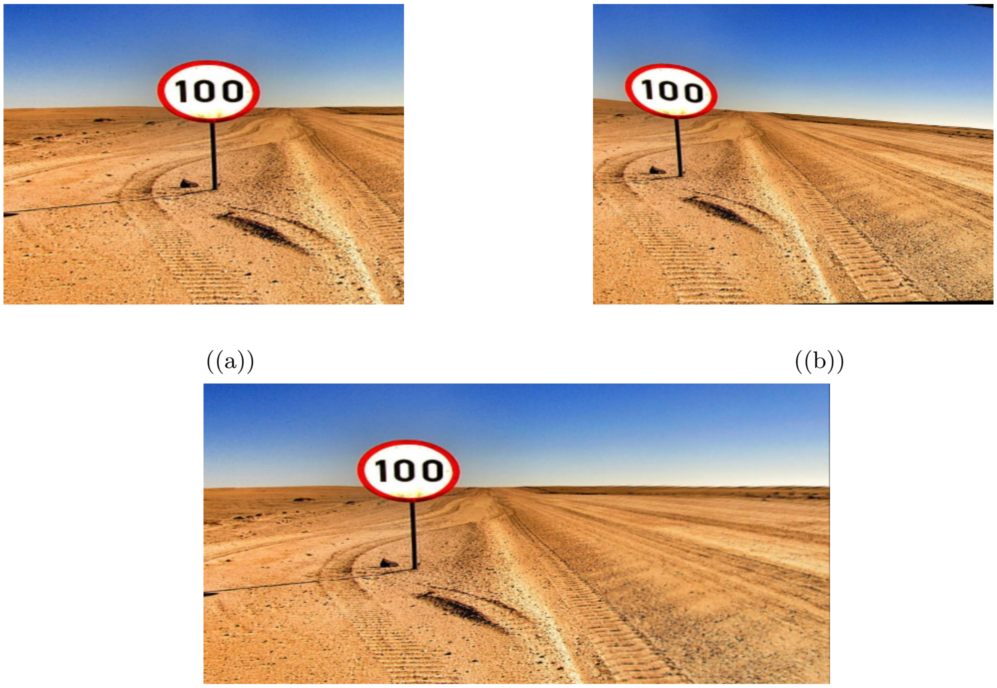
2 Q2 Stitching Images

Stitching multiple images together requires us to identify points of interest in both images that match one another. Using an algorithm such as SIFT that is found in openCV, we can identify n such points of interest in each image. Using the descriptors of each of these points we can attempt to match the points of the 2 images

```
sift = cv.xfeatures2d.SIFT_create()
kp_a, des_a = sift.detectAndCompute(gray_a,None) ## Points of interest in image A
kp_b, des_b = sift.detectAndCompute(gray_b,None) ## Points of interest in image B
bf = cv.BFMatcher()
matches = bf.knnMatch(des_b,des_a,k=2) ## Finding points that match
```

Listing 1: Snippet to generate and match POI

Using the list of matches found, we can gather the list of points in image A that match image B. We can calculate the homography that moves the points of image B to the points on image A. Using this homography matrix we can warp image B and superimpose image A onto it to get the stitched image. The Pipeline of this process is described in figure 3. The results can be see in figure 4



((a))

((b))

((c))

Figure 4: **Question 2:** 4(a) Image left. 4(b) Image right. 4(c) Final stitched Image

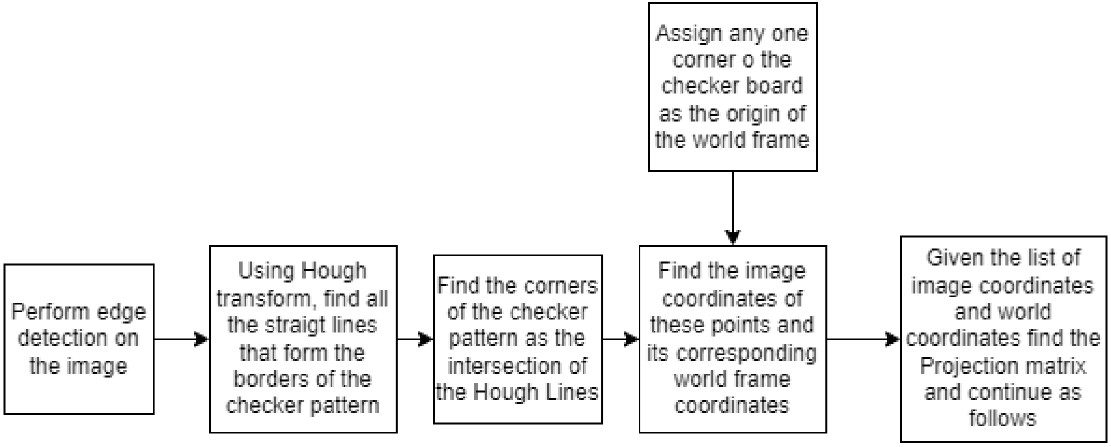
3 Q3 Camera Calibration

3.1 Minimum number matching points

The minimum pair of points (both image coordinate and world coordinate) require to solve for calibration is 6. Using 6 points we can solve for every element in the project matrix (12 elements).

3.2 Pipeline to calibrate

To calibrate a camera using two perpendicular checkerboard patterns are required. We need to know the precise location of the corners in the world frame (whose origin can be taken arbitrarily at any point on the plane). The corresponding image coordinates can be calculate by detecting the corners of the image. Using these two set of points(minimum 6 image points and its corresponding world point) we can construct **A** matrix (refer the following equation3.3). Using the math described in the following section, we can calculate the value of **P** the Projection matrix. Decomposing the Projection Matrix using QR factorization, we can obtain the intrinsic and extrinsic matrices.


Assign any one corner o the checker board as the origin of the world frame
Perform edge detection on the image
Using Hough transform, find all the straigt lines that form the borders of the checker pattern
Find the corners of the checker pattern as the intersection of the Hough Lines
Find the image coordinates of these points and its corresponding world frame coordinates
Given the list of image coordinates and world coordinates find the Projection matrix and continue as follows

3.3 Math Behind Calibration

In the question we are given the points in image coordinate frame u_i, v_i and its corresponding world coordinates X_i, Y_i, Z_i. We know that the relationship between the two is

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{KR[I|0_{3\times3}]}_{\text{Projection Matrix } \mathbf{P}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Where K the intrinsic parameters of the camera times R the Extrinsic parameters (Combination of Rotation and Translation) gives us the Projection matrix P which is a 3×4 matrix. Given a list of

u and X we can rearrange the above equation and write it as follows

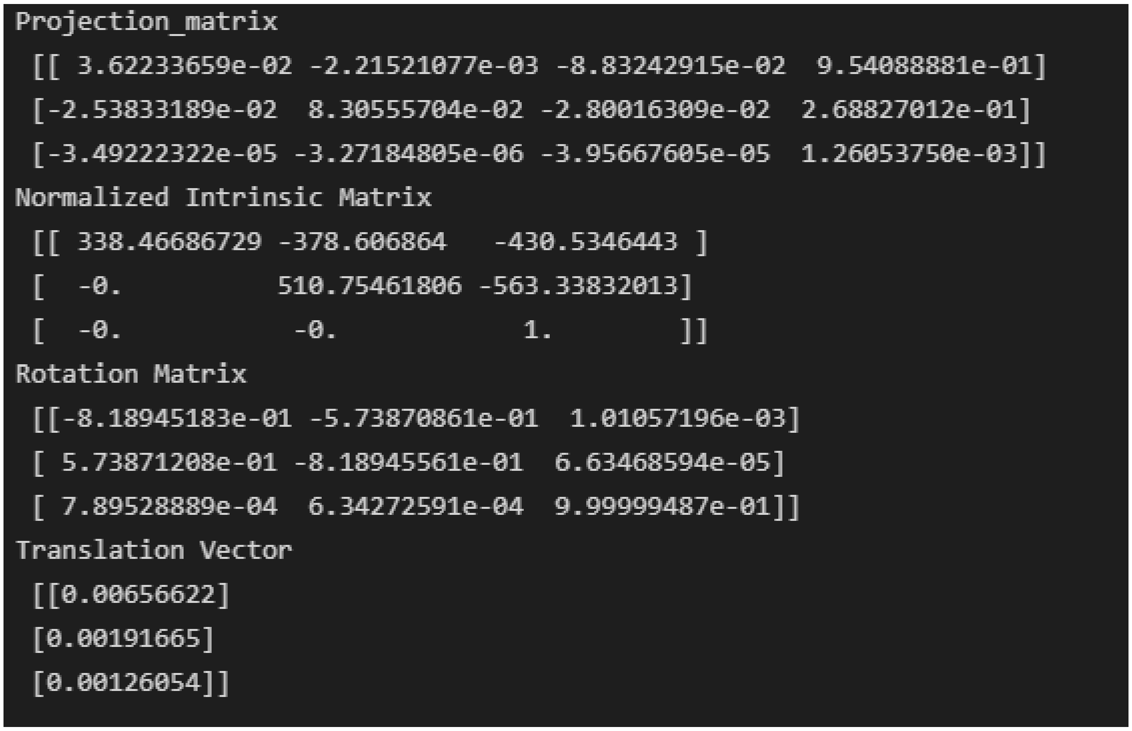
$$\underbrace{\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1x_w^{(1)} & -u_1y_w^{(1)} \\ -u_1z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1x_w^{(1)} & -v_1y_w^{(1)} \\ -v_1z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_ix_w^{(i)} & -u_iy_w^{(i)} \\ -u_iz_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_ix_w^{(i)} & -v_iy_w^{(i)} \\ -v_iz_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_nx_w^{(n)} & -u_ny_w^{(n)} \\ -u_nz_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_nx_w^{(n)} & -v_ny_w^{(n)} \\ -v_nz_w^{(n)} & -v_n \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix}}_{\text{p}} = [0]$$

By solving for the homogeneous equation $Ap = 0$ we can obtain the value of p. But we know that the homogeneous coordinates in the image coordinates are scale invariant, $u = ku$, resulting in $PX = kPX$. This would mean that we need to find the vector p whose square of magnitude is 1 (least square solution of p).

$$\begin{aligned} & ||p||^2 = 1 \\ & \text{Given the two condition } Ap = 0 \text{ and } ||p||^2 = 1 \\ & min_p(p^TA^TAp) \quad \text{such that} \quad p^Tp = 1 \\ & L(p, \lambda) = p^TA^TAp - \lambda(p^Tp - 1) \;\; \text{Where } L(p, \lambda) \text{ is a loss function} \\ & \text{Taking the derivatives off } L(p, lamba) \text{ gets} \\ & A^TAp = \lambda p \end{aligned} \tag{1}$$

Where p be a set of eigen vectors of A^TA, and our solution will be the one that corresponds to the least eigen value.

we can rearrange the corresponding eigen vector to get P which is a 3×4 matrix. We know that the first 3 rows and columns of the matrix is obtained as a product of K and R, an upper right triangular



```
Projection_matrix
 [[ 3.62233659e-02 -2.21521077e-03 -8.83242915e-02  9.54088881e-01]
 [-2.53833189e-02  8.30555704e-02 -2.80016309e-02  2.68827012e-01]
 [-3.49222322e-05 -3.27184805e-06 -3.95667605e-05  1.26053750e-03]]
Normalized Intrinsic Matrix
 [[ 338.46686729 -378.606864   -430.5346443 ]
 [  -0.          510.75461806 -563.33832013]
 [  -0.           -0.            1.        ]]
Rotation Matrix
 [[-8.18945183e-01 -5.73870861e-01  1.01057196e-03]
 [ 5.73871208e-01 -8.18945561e-01  6.63468594e-05]
 [ 7.89528889e-04  6.34272591e-04  9.99999487e-01]]
Translation Vector
 [[0.00656622]
 [0.00191665]
 [0.00126054]]
```

Figure 5: **Question 3:** Final Output

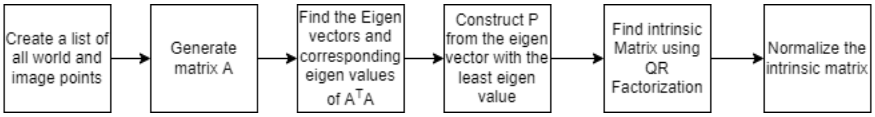

Create a list of all world and image points
Generate matrix A
Find the Eigen vectors and corresponding eigen values of A^TA
Construct P from the eigen vector with the least eigen value
Find intrinsic Matrix using QR Factorization
Normalize the intrinsic matrix

Figure 6: Pipeline for Question 3

matrix and ortho-normal matrix respectively. Hence it can be decomposed using QR factorization, to obtain the individual K and R matricies. The translation vector can be obtained by the following formula, $t = K^{-1}\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$

3.4 Program to calculate the K matrix

As shown in figure 6

- Generate the Matrix A given the list of world and image points
- Calculate the eigen value and eigen vector for A^TA.
- Select the eigen vector corresponding to the least eigen value

- Extract the 3 × 3 matrix from P and perform QR Factorization
- Normalize the intrinsic matrix such that the last element is 1

The results of the above steps are shown in the figure 5

4 Q4 Implement K means

As illustrated in the figure 7

1. Flatten the image into a list of points with 3 feature (RGB values)
2. Randomly select any point in this space to be the starting centroid
3. Assign each point in the set of points to any one centroid based on distance.
4. Calculate the new centroid of each newly created cluster
5. Compare the newly created centroid with the previous one or convergence
6. Repeat from step 3 until converged or max number of iterations are complete
7. Replace each points value in the flattened image with the value of the centroid class it belongs to.
8. Reshape the image back to the original dimensions
9. Display image

Due to random selections of centroid points, the 4 classes can vary as the colors in the image are very close to each other. This is show in the Fig 8. It is noticed that in 8(b) the red lights are grouped into the same class as the body of the traffic lights while a new class for the pole has emerged as compared to 8(c) where the red light is its own color class and the pole has be classified along with the background

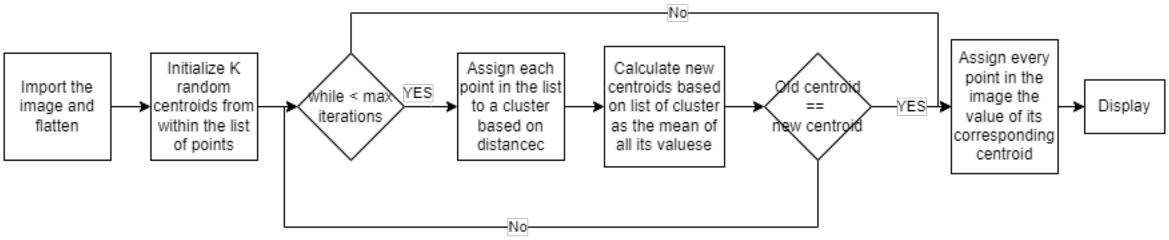
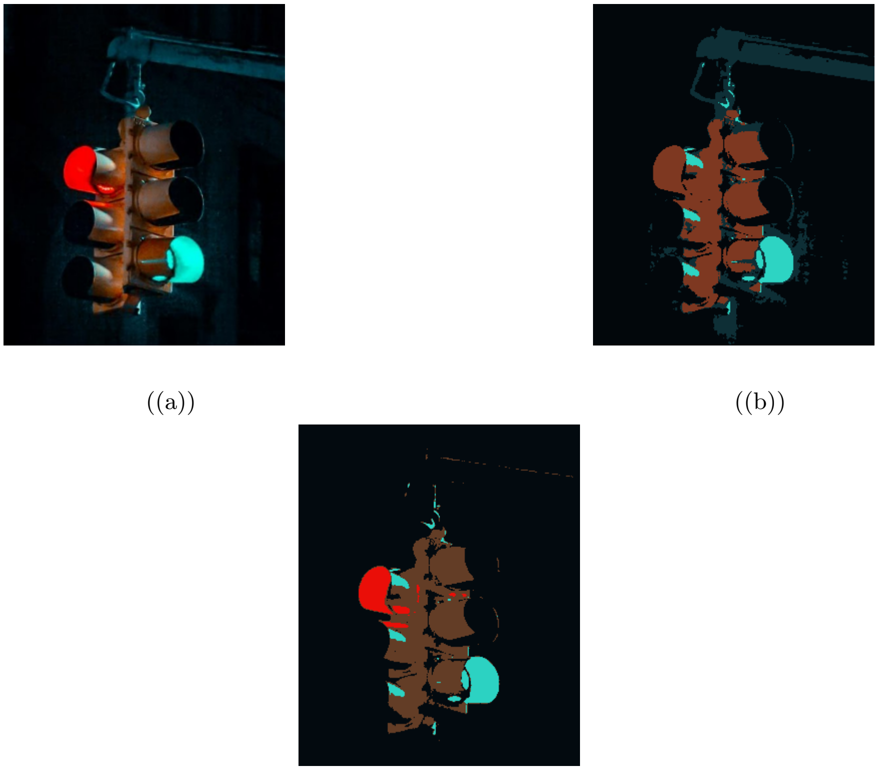

Import the image and flatten
Initialize K random centroids from within the list of points
while < max iterations
YES
Assign each point in the list to a cluster based on distancec
Calculate new centroids based on list of cluster as the mean of all its valuese
Old centroid == new centroid
YES
Assign every point in the image the value of its corresponding centroid
Display
No
No

Figure 7: Pipeline for Question 4



((a))

((b))

((c))

Figure 8: **Question 4:** 8(a) Base image. 8(b) Image segmented with a random seed of 42. 8(c) Image segmented with a random seed of 690420.