ENPM 673

Perception of Robotics

Midterm Exam

By

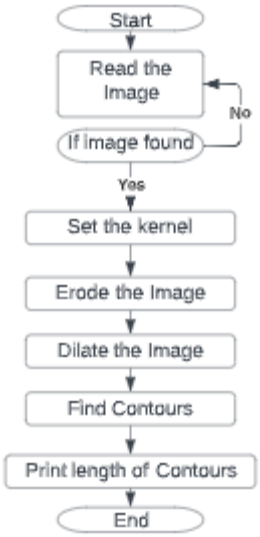
Mothish Raj Venkatesan Kumararaj

Question 1:

Question 1 (20 marks): Assuming the image below represents an x-ray of old coins. Read this image seen below using OpenCV taking into consideration that this is a binary image.

1. Write a program that will separate each coin from each other, so that the output image has the coins completely separated.
2. Write a program that will automatically count how many coins are there in the image (from step 1) . Note: You are allowed to use **any** built-in function in OpenCV. However, you are required to explain each step you took in your solution (Use pipeline or a block diagram).

Pipeline:

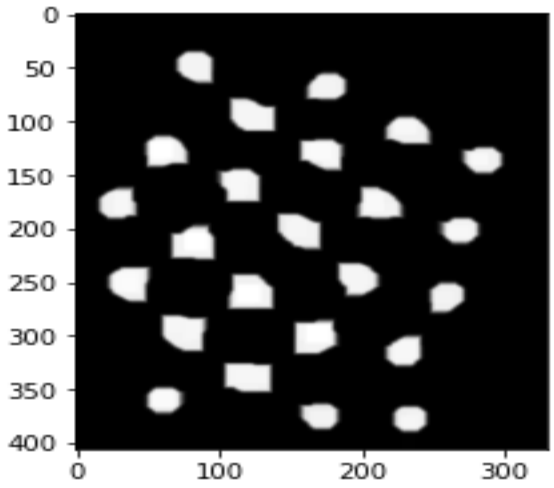


Process:

- Read the given image using openCV.
- Set the kernel for the image morphology.
- Erode the image with 3 iterations to separate the coins.
- Dilate the image to get a clear image of the coins.
- Use findContours to get the number of corners.
- Length of the contour will be the number of coins

Results:

Image After Separation:

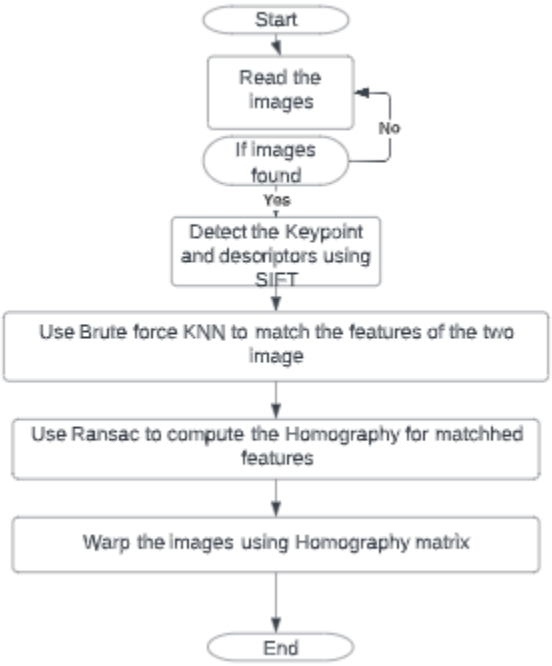


Answer: 24

Question 2:

Question 2: (30 marks) Given two images, imageA and imageB in this link shown below, find the matching points between these two images and stitch them together by finding the homography between them. Write the pipeline that you used to solve this problem and attach the code solution.

Pipeline:

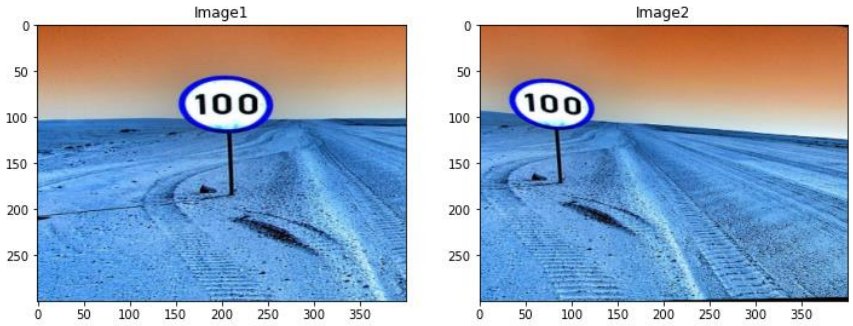


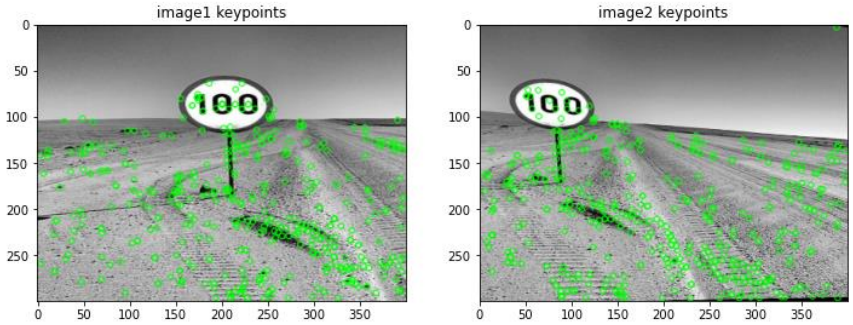
Process:

- Read the given images using openCV.
- Detect the Key points and descriptors of the two images using SIFT.
- Brute force KNN match the descriptors by setting ratio.
- Use RANSAC and compute the homography matrix for the matched features.
- Warp the image using the homography matrix.

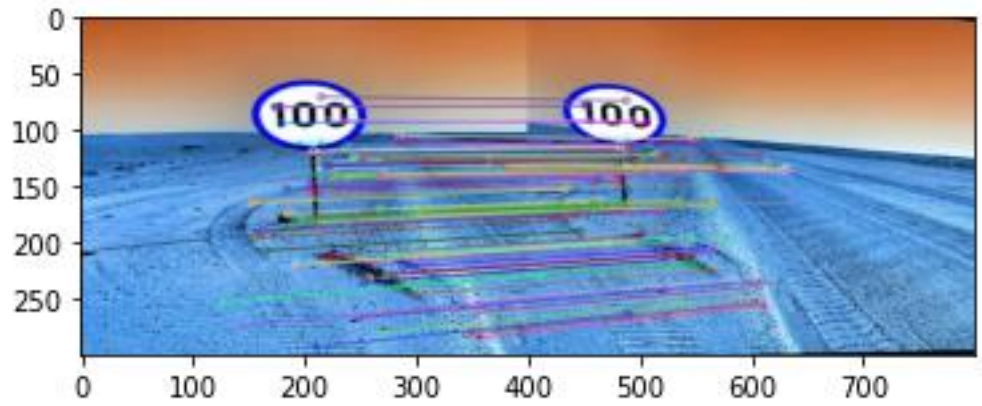
Results:

Key points and Descriptors (Features):





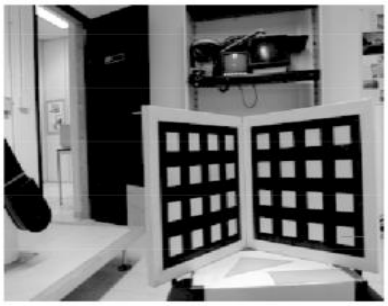
Features Matched:



Stitched Image:

Question 3:

Question 3 (30) Calibrate the camera (Find the intrinsic matrix K).



1. What is the minimum number matching points to solve this mathematically?
2. What is the pipeline or the block diagram that needs to be done in order to calibrate this camera given the image above.
3. First write down the mathematical formation for your answer including steps that needs to be done to find the intrinsic matrix K
4. Write a program that will calibrate the camera given the point correspondences from world to image. Use the data provided below. Please note you are only allowed to use numpy for this question. No marks will be given if you use any other library/tool.

Results:

Question 1: Answer = 6

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} u \\ v \\ w \\ T \end{bmatrix}$$

now we normalize the equation

$$\begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix} = P \begin{bmatrix} U/T \\ V/T \\ W/T \\ 1 \end{bmatrix}$$

now we take

$x_i = U/w$ (image co-ordinate)

$y_i = V/w$ (image co-ordinate)

$X_w = U/T$ (world)

$Y_w = V/T$ (world)

$Z_w = W/T$ (world)

$$x_1 = \frac{p_{11} x_w + p_{12} y_w + p_{13} z_w + p_{14}}{p_{31} x_w + p_{32} y_w + p_{33} z_w + p_{34}}$$

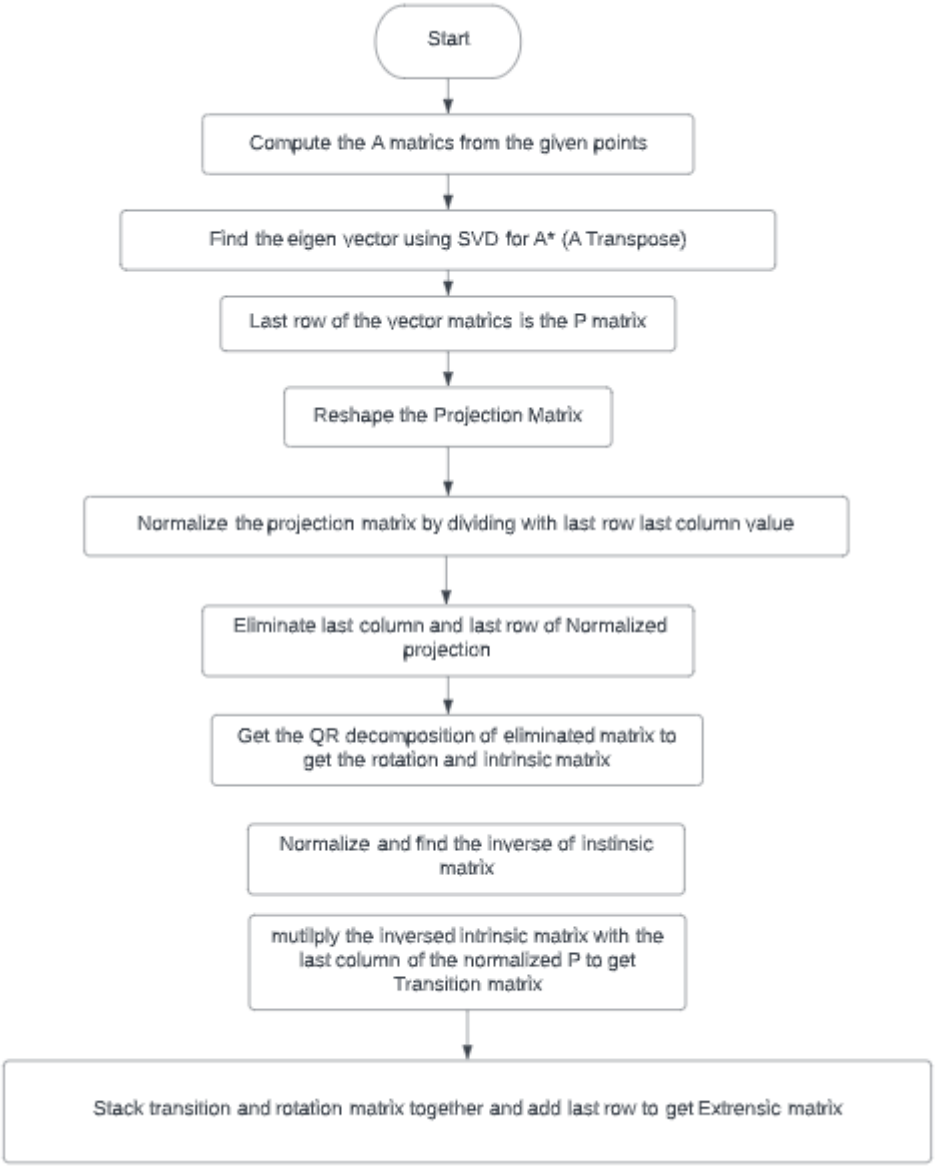
$$y_1 = \frac{p_{21} x_w + p_{22} y_w + p_{23} z_w + p_{24}}{p_{31} x_w + p_{32} y_w + p_{33} z_w + p_{34}}$$

since we have 11 unknown in p we get two equation for each point

$$\therefore \frac{11}{2} = 5.5 \simeq 6$$

$\therefore$ we need 6 points

Question 2 :


Start
Compute the A matrics from the given points
Find the eigen vector using SVD for A* (A Transpose)
Last row of the vector matrics is the P matrix
Reshape the Projection Matrix
Normalize the projection matrix by dividing with last row last column value
Eliminate last column and last row of Normalized projection
Get the QR decomposition of eliminated matrix to get the rotation and intrinsic matrix
Normalize and find the inverse of instinsic matrix
mutilply the inversed intrinsic matrix with the last column of the normalized P to get Transition matrix
Stack transition and rotation matrix together and add last row to get Extrensic matrix

Question 3:

$$u = m_x f \frac{x_c}{z_c} + o_x \qquad v = m_y f \frac{y_c}{z_c} + o_y$$

Where mx and my is the pixel densities and Xc and Yc are camera frame coordinates. Oy and Ox are the distance from the principal point and f is a focal length. U and V are the image sensor coordinates

The homogenous representation of 2D points in 3D array is given by

$$\mathbf{u} \equiv \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{w}u \\ \tilde{w}v \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \tilde{\mathbf{u}}$$

Where,

$$u = \frac{\tilde{u}}{\tilde{w}} \qquad v = \frac{\tilde{v}}{\tilde{w}}$$

From the given equation the intrinsic matrix can be obtained:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Intrinsic Matrix:

$$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

Now that perspective projection from camera coordinate to image is done and we need to get extrinsic matrix

$$\tilde{\mathbf{x}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Where Xc is camera coordinate and Xw is world coordinates.

We get the rotational and transitional matrix combined to form extrinsic matrix

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

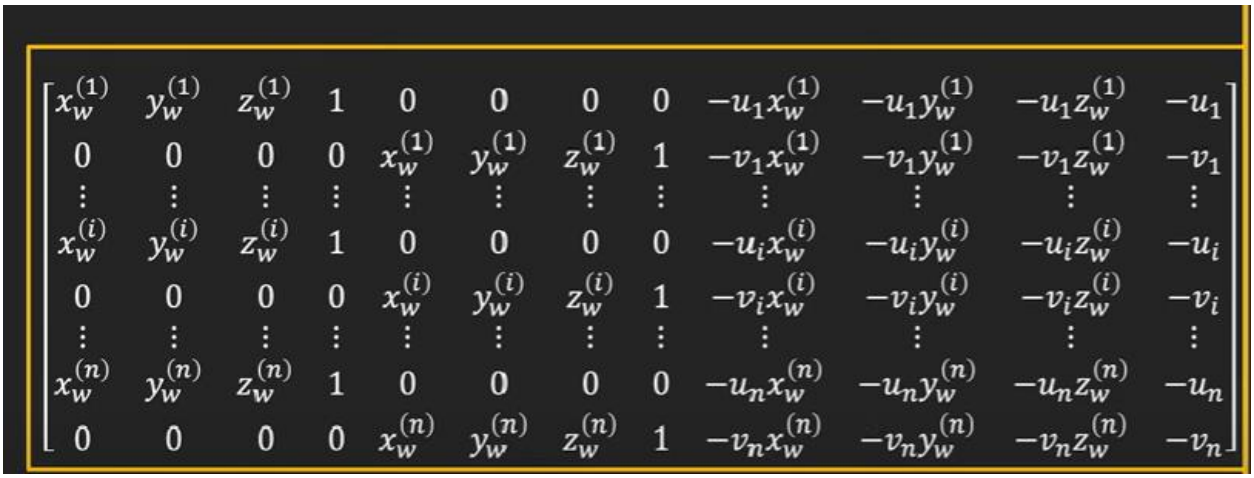
Now to get the world to image we take intrinsic and extrinsic matrix

$$\tilde{\mathbf{u}} = M_{int} M_{ext} \, \tilde{\mathbf{x}}_w = P \, \tilde{\mathbf{x}}_w$$

Where Mint is intrinsic and Mext is extrinsic and with the help of that we get projection matrix

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

And we can get A matrix from



$$\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1 x_w^{(1)} & -u_1 y_w^{(1)} & -u_1 z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1 x_w^{(1)} & -v_1 y_w^{(1)} & -v_1 z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_i x_w^{(i)} & -u_i y_w^{(i)} & -u_i z_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_i x_w^{(i)} & -v_i y_w^{(i)} & -v_i z_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_n x_w^{(n)} & -u_n y_w^{(n)} & -u_n z_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_n x_w^{(n)} & -v_n y_w^{(n)} & -v_n z_w^{(n)} & -v_n \end{bmatrix}$$

Then we follow steps from questions 2

Question 4:

Projection:

0.0362234	-0.00221521	-0.0883243	0.954089
-0.0253833	0.0830556	-0.0280016	0.268827
-3.49222e-05	-3.27185e-06	-3.95668e-05	0.00126054

Extrinsic:

-0.818945	-0.573871	0.00101057	-50.2456
0.573871	-0.818946	6.63469e-05	-14.6664
0.000789529	0.000634273	0.999999	-9.64579
0	0	0	1

Intrinsic:

-35.0896	39.251	44.6344
0	-52.951	58.4025
0	0	-0.103672

Question 4:

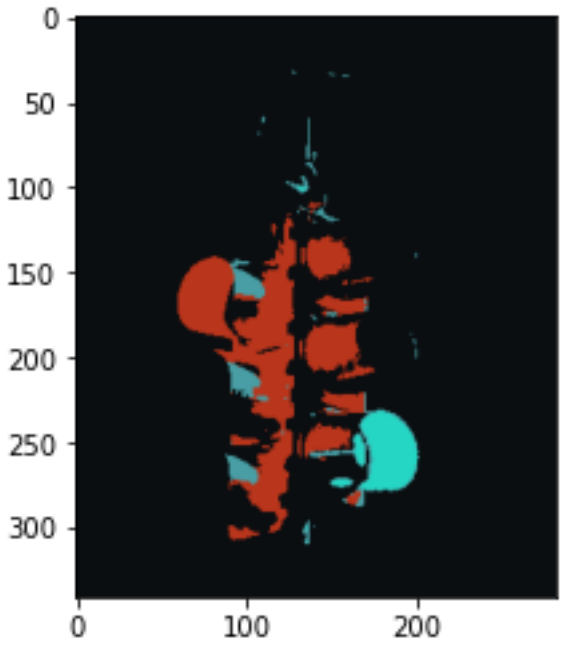
Question 4: (20 marks) Implement K-means algorithm to separate this image below, based on color, into 4 classes. Note: You are NOT allowed to use any built-in function, implement your code from scratch. Explain each step you take.

Process:

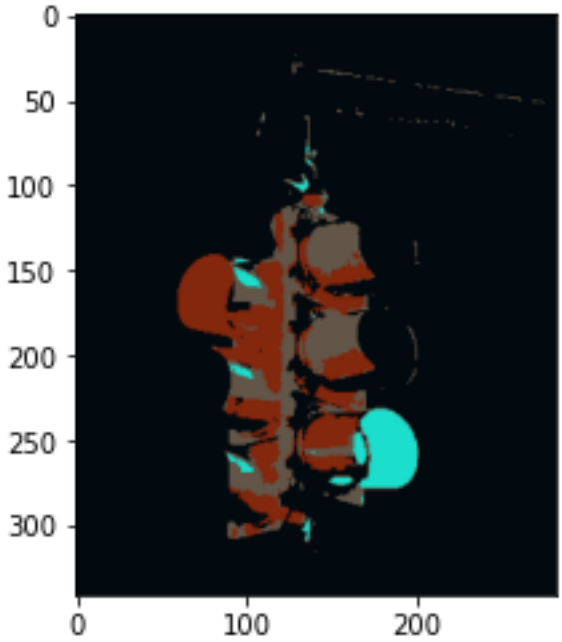
- Read the given image using openCV.
- Create 4 centroid points based on the color to be filtered (Red, Green, Blue and Black in our case).
- Iterate through each pixel in the image and get the Euclidean distance between the pixel and the centroid of each color
- Find the minimum of the distances and append the pixel into the clusters of respective color.
- Find the average of each cluster and take that as the new centroid points.
- If the Euclidean distance between the previous and current centroid is less than .7 the break.
- Change each pixel with the closest centroid points to get the segmented image.

Results:

Iteration 1:



Iteration 20:



Iteration with 0.7 threshold: