Project – Report

Omkar. A. Chittar

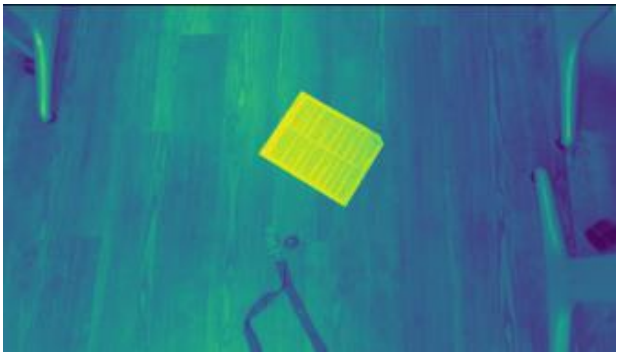
119193556

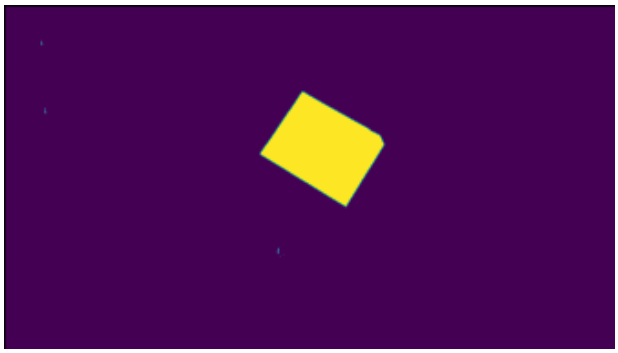
Q 1. In this problem, you will perform camera pose estimation using homography. Given this video your task is to compute the rotation and translation between the camera and a coordinate frame whose origin is located on any one corner of the sheet of paper. In order to do so, you must:
Design an image processing pipeline to extract the paper on the ground and then extract all of its corners using the Hough Transformation technique.
Once you have all the corner points, you will have to compute homography between real world points and pixel coordinates of the corners. You must write your own function to compute homography.
Decompose the obtained homography matrix to get the rotation and translation.
Note: If you decide to resize the image frames, you need to accordingly modify your intrinsic matrix too. Refer to this discussion.

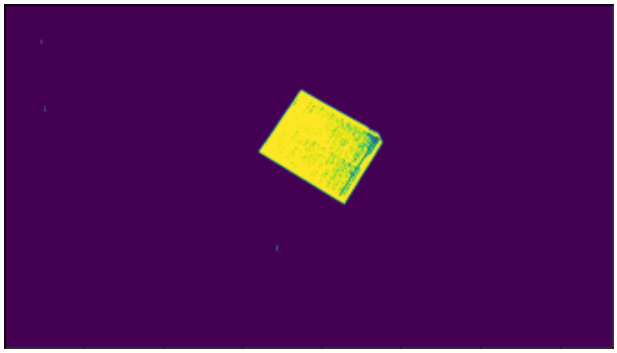
Data: The dimensions of the paper are 21.6 cm x 27.9 cm.
The intrinsic matrix of the camera can be found here.

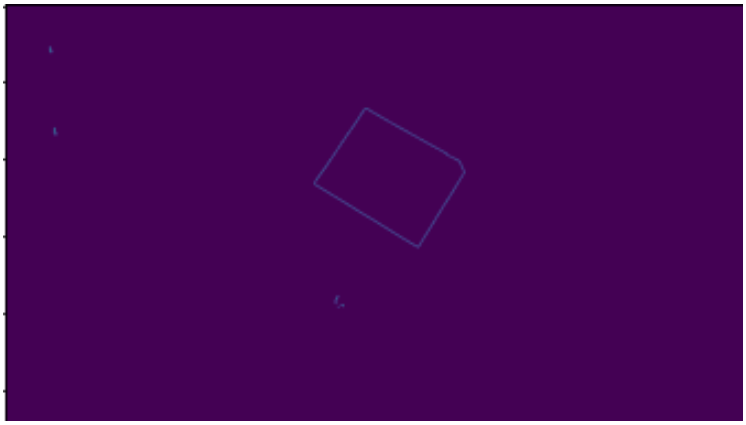
Approach:

1. Import the req libraries - **numpy**, **OpenCV**, and **matplotlib**.
2. Define a function named '**houghTransform**' that takes two arguments, an **image** and a threshold.
3. Inside the function, perform the following image preprocessing steps:
   a. Convert the image to grayscale.
   b. Apply a Gaussian blur to the image.
   c. Threshold the image using a specific threshold value.
   d. Apply edge detection using Canny edge detector.



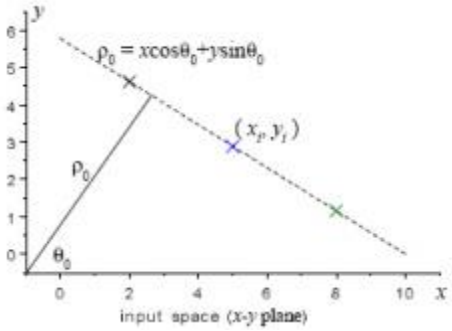


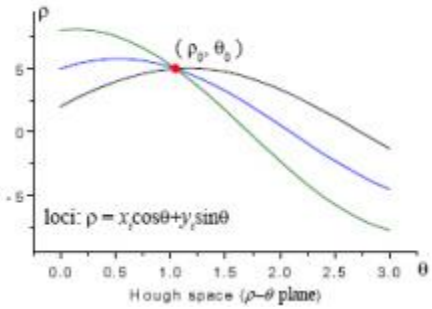




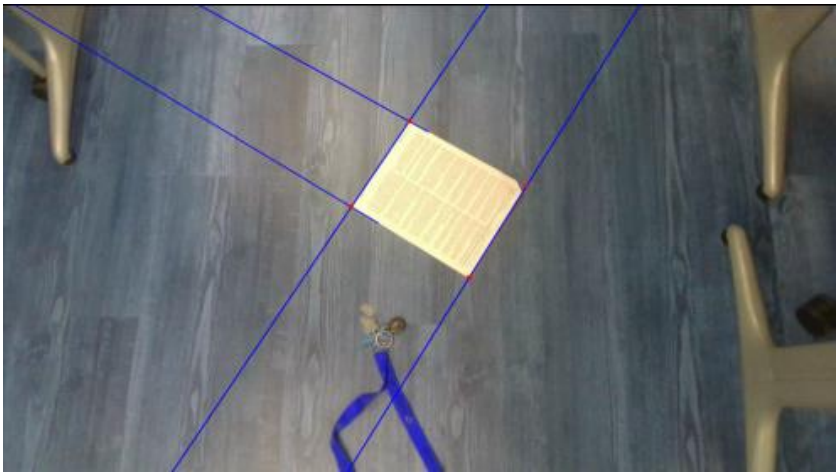
Define Hough transform parameters – theta_values and d_values.

4. Create an accumulator array of zeros with the shape of (len(**d_values**), len(**theta_values**))
5. Loop over all edge pixels and vote for every possible line using Hough transform:
   a. Compute the corresponding value of d for every theta value.
   b. Find the nearest d value from the array of d values.
   c. Increment the accumulator array at the corresponding (d, theta) location.



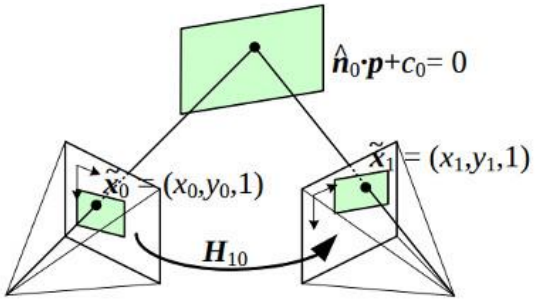


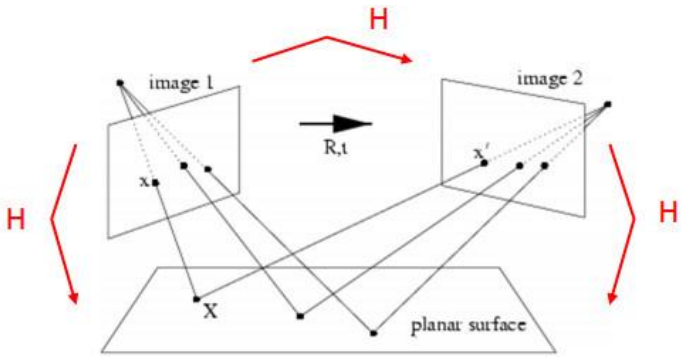
6. Extract the peaks from the accumulator array that exceed the specified threshold value.
7. Find the intersections of all possible lines represented by the detected peaks.
8. Sort the intersections based on x and y coordinates.
9. Select the corners by choosing the top-left, top-right, bottom-left, and bottom-right intersections.
10. Print the detected corners and return them.



11. Define a function named 'homography' that takes two arguments, source points and destination points.
12. Check that both input arrays have the same shape and contain at least four points.
13. Construct the A matrix using source and destination points.

14. Implement RANSAC algorithm to estimate the homography matrix that best fits the source and destination points.





$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera Intrinsic Camera Extrinsic

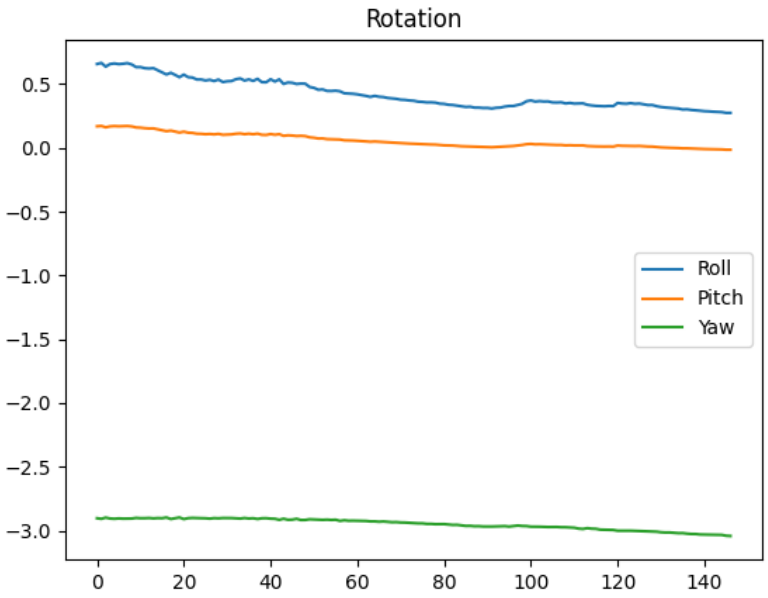
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

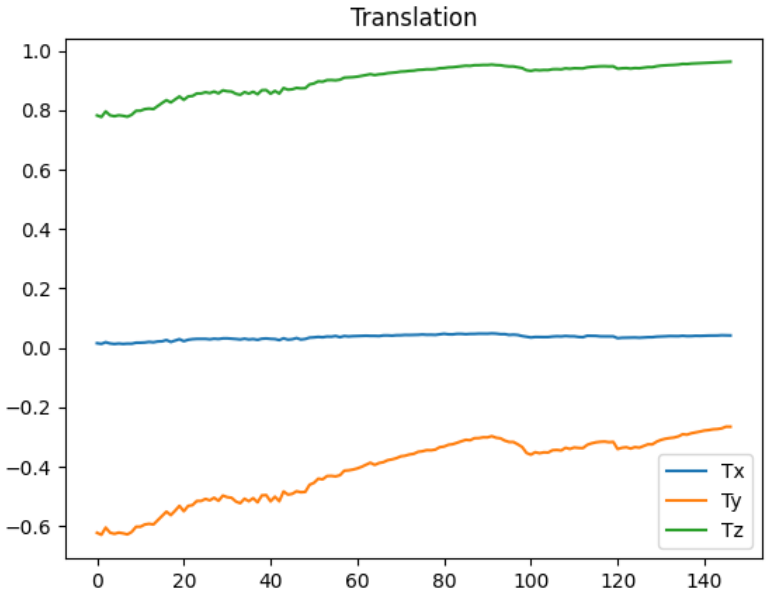
15. Randomly select four points correspondences and calculate the homography matrix.
16. Calculate the number of inliers using the distance between predicted and actual destination points.
17. Repeat steps 15-16 for n iterations and select the homography matrix with the maximum number of inliers.
18. Calculate the homography matrix using all inliers and return it.
19. Calculate the rotation and translation matrices for each frame using this Homography matrix.
20. Calculate Roll, Pitch and Yaw values from the obtained rotation matrix. The Translation values don't require any computation.

```
def rotations(R):
    pitch = -np.arcsin(R[2][0])
    roll = np.arctan2(R[2][1]/np.cos(pitch), R[2][2]/np.cos(pitch))
    yaw = np.arctan2(R[1][0]/np.cos(pitch), R[0][0]/np.cos(pitch))
    return roll, pitch, yaw
```

21. Plot the roll, pitch, and yaw angles and the X, Y, and Z translations as functions of time using matplotlib.

Results:





Problems Encountered:

- Getting exactly 4 points in all the detected frames took a lot of time and thresholding.
- The code for performing ‘hough transform’ takes a lot of time to execute. So, I ran the code once and then copied all the corner values into a list which can be used to find out the homography directly.