ENPM673

Perception for Autonomous Robots — PROJECT-2

22 February 2023

Vamshi Kalavagunta

119126332

vamshik@umd.edu

A. James Clark School of Engineering

University of Maryland College Park - 20742

1 Problem-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:

1. Design an image processing pipeline to extract the paper on the ground and then extract all of its corners using the Hough Transformation technique .
2. 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.
3. Decompose the obtained homography matrix to get the rotation and translation

1.1 Solution - pipeline

Here are the steps involved in camera pose estimation using homography:

1. Design an image processing pipeline to extract the paper on the ground and then extract all of its corners using the Hough Transformation technique.
2. Detect the edges of the paper using canny edge detection and get the slope and intercepts and extract the corners.
3. Once you have all the corner points, compute homography between real world points and pixel coordinates of the corners.
4. Decompose the obtained homography matrix to get the rotation and translation.
5. Apply the obtained rotation and translation to the coordinate frame whose origin is located on any one corner of the sheet of paper.
6. Repeat steps 1-4 for each frame in the video.
7. Display the resulting frames with the estimated camera pose.

1.2 Hough Transform

The Hough transform is a technique used in computer vision and image processing to detect shapes, such as lines or circles, in digital images. It works by converting points in the image to a parameter space, where each point represents a possible line or circle. The transform produces a set of curves, each representing a possible line or circle in the image.

Basic Algorithm steps for Hough transform is :
Extract edges of the image using Canny

1. initialize parameter space rs, thetas
2. Create accumulator array and initialize to zero
3. for each edge pixel
4. for each theta
5. calculate r = x cos(theta) + y sin(theta)
6. Increment accumulator at r, theta
7. Find Maximum values in accumulator (lines)
8. Extract related r, theta

1.3 Homography

In computer vision, the concept of homography is used to analyze, explain, and study visual perspective, especially the difference in appearance of two flat objects viewed from different points of view.the homography between them is computed using the following system of equations $Ax = 0$, where A is given by

$$A = \begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1*xp1 & y1*xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1*yp1 & y1*yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2*xp2 & y2*xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2*yp2 & y2*yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3*xp3 & y3*xp3 & xp3 \\ 0 & 0 & 0 & -x3 & -y3 & -1 & x3*yp3 & y3*yp3 & yp3 \\ -xn & -yn & -1 & 0 & 0 & 0 & xn*xpn & yn*xpn & xpn \\ 0 & 0 & 0 & -xn & -yn & -1 & xn*ypn & yn*ypn & ypn \end{bmatrix}, X = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix}$$

Singular Value decomposition is a generalized version of eigen decomposition where we factorize an mxn matrix A of rank r as $A = U\Sigma V^T$

Here

1. columns of U are orthogonal eigenvectors of AA^T. It is an mxm square matrix.
2. columns of V are orthogonal eigenvectors of A^TA. It is an nxn square matrix.
3. $\Sigma = diag\,(\sigma_1, \sigma_2 \ldots)$. Where $\sigma = \sqrt{\lambda}$ and λ is the eigen value of AA^T or A^TA. Σ is an $m \times n$ diagonal rectangular matrix which acts as a scaling matrix.
4. When we compute the SVD of a matrix, we divide it into three parts: U , $Sigma$, and V. To implement SVD, A must be a positive semi-definite matrix, which means that A must equal 0 or all eigenvalues must be non-negative.

$$A = U\Sigma V^T$$

5. In this scenario, input to the matrix implies being multiplied by a vector on the right. When we have an input to A that is in the direction of the nth column of V, it will be scaled by the nth value of $Sigma$ and output as the nth column of U. Consequently, when the scaling factor matrix $Sigma$ is sorted, the final column of V determines the input direction of A with the least output. As a result, that is the direction of input to A that will result in the output being closest to zero and is the vector that will minimize Ah. We minimize $|Ah|$ subject to ——$h| = 1$. h would be the final value.

$$A = U\Sigma V^T, A = \left(AA^T\right)(\Sigma)\left(A^TA\right)^T$$

1.4 Camera pose Estimation

Assume all points lie in one line with z=0

$$\boldsymbol{X} = (X, Y, 0, 1)$$

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{P}\boldsymbol{X} \\ &= \boldsymbol{K}[\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{r}_3\boldsymbol{t}] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \\ &= \boldsymbol{K}[\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{t}] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \\ &= \boldsymbol{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \end{aligned}$$

$$\boldsymbol{H} = \lambda \boldsymbol{K}[\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{t}]$$

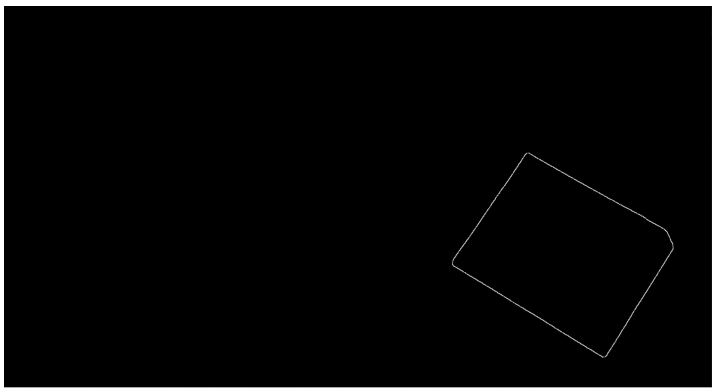
$$\boldsymbol{K}^{-1}\boldsymbol{H} = \lambda[\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{t}]$$

–r_1 and r_2 are unit vectors ⇨ find lambda
–Use this to compute t
–Rotation matrices are orthogonal ⇨ find r3

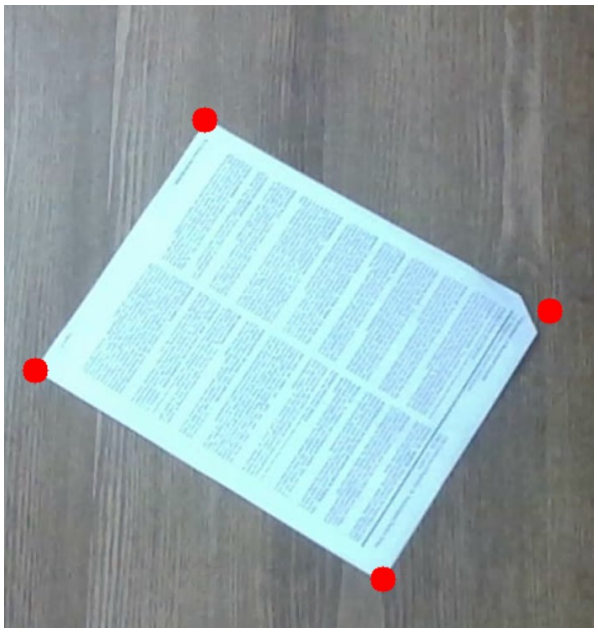
$$\boldsymbol{P} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & (\boldsymbol{r}_1 \times \boldsymbol{r}_2) & \boldsymbol{t} \end{bmatrix}$$

1.5 Results

Edge detection



Corner detection rotation and translation



2 Problem-2

You are given four images which were taken from the same camera position (only rotation no translation) you will need to stitch these images to create a panoramic image.

To solve this problem, you will need to:

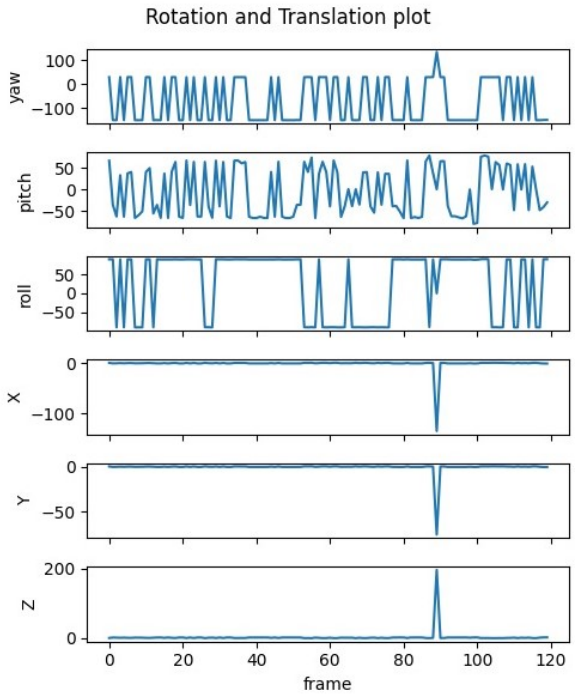
1. Extract features from each frame (You can use any feature extractor).
2. Match the features between each consecutive image and visualize them.
3. Compute the homographies between the pairs of images
4. Combine these frames together using the computed homographies.

2.1 Solution - Pipeline

The implementation of image stitching, which is the process of combining multiple images with overlapping fields of view to produce a single panoramic or high-resolution image.

The code performs the following steps:

1. Load four input images image1, image2, image3, and image4.
2. Call the "getpoints" function to extract SIFT key-points and descriptors from pairs of adjacent images (image1 and image2, image3 and image4).
3. Call the "homography" function to compute the homography matrix for each pair of adjacent images.
4. Use the homography matrices to warp the images and stitch them together. Specifically, the code warps image2 to align with image1 using H1, and warps image4 to align with image3 using H2. The warped images are then cropped and stitched together. The resulting stitched image is displayed.



The "getpoints" function extracts SIFT keypoints and descriptors from two input images and uses a brute force matcher to find pairs of matching keypoints. The matches are filtered using the ratio test to eliminate ambiguous matches. Finally, the function returns a set of point correspondences between the two images.

The "computehomography" function takes a set of four point correspondences between two images and computes the homography matrix that maps points from one image to the other. The homography matrix is computed using SVD.

The "distance" function takes a point correspondence and a homography matrix and computes the geometric distance between the projected point in one image and the corresponding point in the other image. This distance is used in the RANSAC algorithm to determine the quality of a set of point correspondences.

The "ransac" function implements the RANSAC algorithm to find the best set of point correspondences that fit a homography matrix. The function randomly selects sets of four point correspondences, computes the homography matrix using the "homography" function, and evaluates the quality of the homography using the "distance" function. The function returns the best homography matrix based on the set of point correspondences that produce the lowest average distance

2.2 Results

2.3 Problems encountered

1. problem during cropping the stitched image. 2. images size was not same it was difficult to stitch.