Spring 2023

Project 2

Perception for autonomous robots

March 8 2023

Student:
Rishikesh Jadhav

Instructors:
Dr. Samer Charifa

Course code:
ENPM 673

Contents

1 Question1

In this question, we are tasked with performing camera pose estimation using homography on a video. The goal is to compute the rotation and translation between the camera and a coordinate frame whose origin is located on any one corner of a sheet of paper.

To achieve this, we need to design an image processing pipeline to extract the paper from the ground and then extract all its corners using the Hough Transformation technique. After obtaining all the corner points, we need to compute homography between real-world points and pixel coordinates of the corners. It is required to write our own function to compute homography.

Finally, we need to decompose the obtained homography matrix to obtain the rotation and translation between the camera and the coordinate frame. This entire process involves multiple steps of image processing, homography computation, and matrix decomposition.

1.1 Image Processing Pipeline

1. Read video frame by frame
2. Grayscale the image.
3. Blur the image.
4. Apply Thresholding to extract white colour.
5. Perform canny edge detection.
6. Use Hough transform function algorithm on given frame.
7. Find peaks in the Hough space.
8. Draw lines corresponding to the Hough peaks on the input image.
9. Find the intersections between the detected lines.
10. Compute the homography matrix between the camera and the ground plane.
11. With all of these find the rotation and translation of the camera.

1.2 Explanation and Results

The homography equation is used to describe the transformation between two images taken from different viewpoints or perspectives. Suppose we have two images, the first image coordinates are denoted by $\mathbf{x} = [x, y, 1]^T$ and the second image coordinates are denoted by $\mathbf{x}' = [x', y', 1]^T$, then the homography matrix $\mathbf{H}$ can be expressed as:

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \tag{1}$$

To obtain the camera Pose using Homography. The following steps are used:

1. Hough Transformation technique for corner detection: Define the parameter space: In the case of detecting lines, the parameter space is a 2D array where each element represents a line in polar coordinates, (ρ, θ).
$$H(\rho, \theta) = 0 \tag{2}$$
.

2. Homography computation:
$$H \cdot p = p' \tag{3}$$
where H is the homography matrix, p is a 3D point in real-world coordinates, and p' is its corresponding pixel coordinates in the image. This equation can be solved using linear algebra methods, such as SVD.

3. Homography decomposition:
$$H = K[R|t] \tag{4}$$
where K is the intrinsic camera matrix, R is the rotation matrix, and t is the translation vector. This equation can be obtained by decomposing the homography matrix using the SVD decomposition.

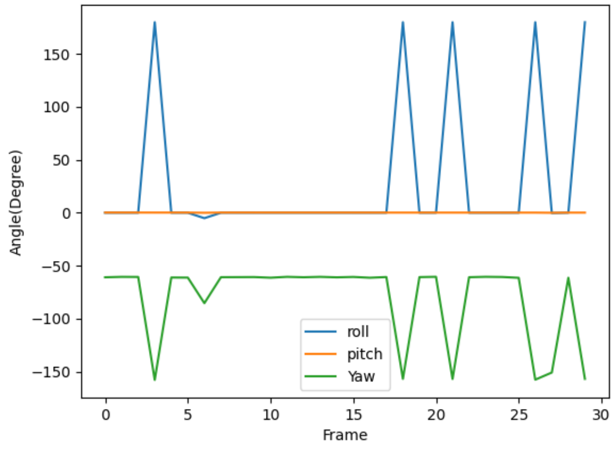

Figure 1: Rotation.

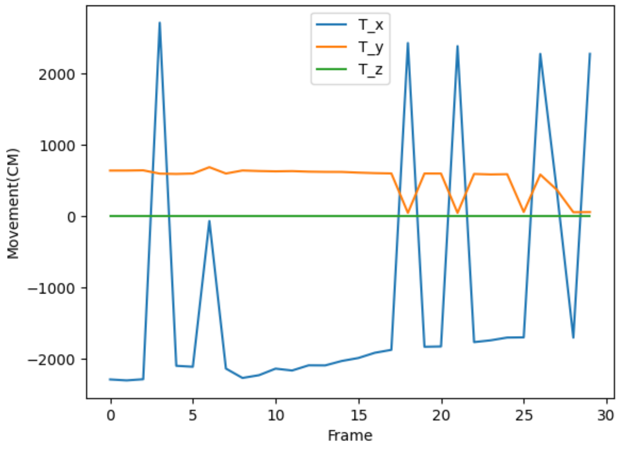

Figure 2: Translation.

2 Question2

In this question, we are given four images taken from the same camera position, but with different angles of rotation. The task is to stitch these images together to create a panoramic image.

To solve this problem, we need to extract features from each image using a feature extractor. Then, we need to match the features between each consecutive image and visualize them to ensure that the matching is correct.

Next, we need to compute homographies between pairs of images. Homographies describe the transformation between two images that are related by a projective transformation. These homographies allow us to map the coordinates from one image to another, so we can stitch them together.

Finally, we need to combine the frames using the computed homographies to create a panoramic image. This involves warping the images according to the homographies and blending them together to create a seamless panorama. This entire process involves feature extraction, feature matching, homography computation, and image blending.

2.1 Pipeline

1. Load the four input images.
2. Convert the images to grayscale to reduce computation time.
3. Extract features from each image using a feature extractor, ORB or SIFT
4. Match the features between each consecutive image using a matching algorithm - Brute-Force Matcher.
5. Visualize the matched features to check their accuracy.
6. Use the matched feature points to compute the homographies between the pairs of images..
7. Compute Homographies between pairs of images.
8. Combine the images together using the computed homographies
9. Warp the second image onto the first image using the computed homography matrix. This step can be done using the OpenCV function warpPerspective.
10. Repeat steps for the next pair of images until all four images have been stitched together.
11. Save the final panoramic image.

2.2 Explanation and Results

To create a panorama from multiple images, we first convert the images to grayscale and use a feature extractor and matcher to obtain corresponding points between the images. These points are used in the RANSAC homography function to obtain a homography matrix that maps the points in one image to the corresponding points in the other image. The RANSAC homography function selects a random subset of points and uses linear algebra to compute the homography matrix. The function then determines which points are inliers and iteratively repeats the process to find the homography with the largest number of inliers. Finally, the homography is re-estimated using all inliers.

The linear algebra involved in the RANSAC homography function aims to find a non-zero solution for the homography matrix that minimizes the sum of squared errors of the residual distances between the transformed points and their true locations. This is done by using singular value decomposition (SVD) to find the nullspace of the matrix of corresponding points and their transformed points. The resulting vector is reshaped into a 3×3 matrix representing the homography.

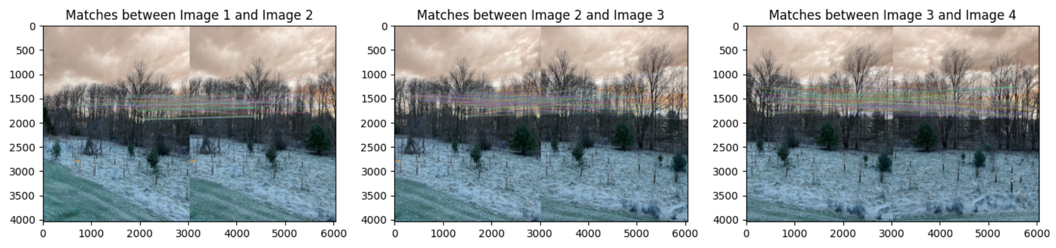
Matches between Image 1 and Image 2
Matches between Image 2 and Image 3
Matches between Image 3 and Image 4

Figure 3: Matching Features Visualization

3 Problems Encountered and solutions

Following road blocks were found during the entire process of completing the project:

1. 1 - In the first question finding the canny function values for edges was a bit tough, after multiple iterations I reached an accurate value.
2. 1.2 - To understand various algorithms and convert them in the python code format was bit tough as I am not very good with the language. I had to read thorough the numpy documentation a lot.
3. 1.3 - Detecting edges was difficult as inbuil functions were not allowed for that
4. 1.4 - finding corners was straightforward after detecing edges correctly
5. 1.5 - Finding the rotation and translation of the camera was a bit tricky.
6. 2.1 -Selecting a feature extractor was tricky between Orb and SIFT
7. 2.2 - Finding matches using Brute force matcher and visualizatiobn was straighforward but writing a function for homography and using it was difficult.
8. 2.3 -Stitching wa the toughest as there were problems of dimensions not being the same and using the Homography on the images to stich was challenging as well.
9. 2.4 - I had to look at various examples to understand it and then execute it. I organised everything using functions, to make things simpler.
10. 2.5 - Taking care of the shape of the various arrays while various computations was difficult.