A Report for Assignment -2 of COL780
Computer Vision

On

Implementation of Harris Corner Detection
and
Image Stitching using Affine Model

Submitted to

Anurag Mittal

(Instructor COL 780)

by

Nitesh Dohre (2022MCS2070)

(Y=0)

COL780 Assignment2

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Session: 2022-23

CONTENTS:

0. PROBLEM STATEMENT

1. Please implement (for Y=0) the Harris Corner Detector. If you feel the need to smooth the image before computing these, you can use an off-the-shelf downloaded or library Gaussian smoother.
2. Then, in adjacent frames, match them using proximity and a simple sum-of-squared-difference approach.
3. Finally, use an affine model to stitch the frames together to create a panorama.

1. INTRODUCTION

Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. In this work, we present an implementation of the Harris corner detector. This feature detector relies on the analysis of the eigenvalues of the autocorrelation matrix. The algorithm comprises several steps, including several measures for the classification of corners, a generic non-maximum suppression method for selecting interest points, and the possibility to obtain the corner's position with subpixel accuracy.

A corner is a point whose local neighbourhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are the important features in the image, and they have generally termed interest points which are invariant to translation, rotation, and illumination.

In the second part, we are asked to find the correspondence points.
Correspondence points refer to matching points or features between two or more images that represent the same physical point or object in the real world. In other words, correspondence points are pairs of points in different images that correspond to the same underlying object or feature.

In the third part, we are asked to stitch multiple images to create a smooth Panorama

2. TOOLS AND TECHNOLOGY USED:

1. Jupyter notebook IDE
2. Python3
3. Libraries: numpy, matplotlib, cv2, maths, os, glob
4. Datasets provided

3. HARRIS CORNER DETECTION

3.1 Algorithm

1. Compute the horizontal and vertical derivatives of the input image using the Sobel filter.
2. Compute the products of the derivatives at each pixel to get the elements of the structure tensor.
3. Apply a Gaussian filter to the elements of the structure tensor to smooth out noise and small variations in intensity.
4. Compute the Harris corner response at each pixel using the smoothed elements of the structure tensor and a constant parameter k.
5. Find local maxima in the Harris corner response using dilation and thresholding. Only select corners whose response is above a certain threshold and is the maximum in its local neighbourhood.
6. Return the coordinates of the detected corners.

3.2 Implementation:

***Finding convolution**: The `Gradient` function takes two parameters `image` and `Sobel`. It appears to be performing convolution of the `image` with a Sobel filter, which is a common technique for computing the gradient of an image.*

The function initializes an empty matrix called intensity with the same dimensions as the image. It then loops through each pixel of the image and performs convolution of the Sobel filter with the 3x3 patch centred around that pixel. The resulting value is then stored in the corresponding location in the intensity matrix.

Sobel Operator:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Harris Matrix:

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

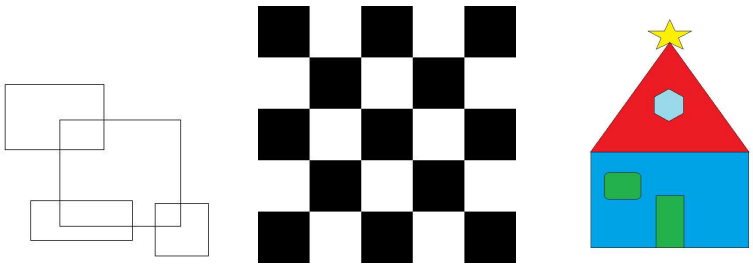
In our code HarrisCornerDetection(img) is implementing the first four steps of the algorithm.

***Non-Maximal Suppression*:** It sorts the corner point in reverse order using the cornerness measure. II. It selects the top of the list as a corner and removes all other corners in its neighbourhood from the list. III. If the Euclidean distance between the points is less than or equal to the window size, then it is considered in the neighbourhood.

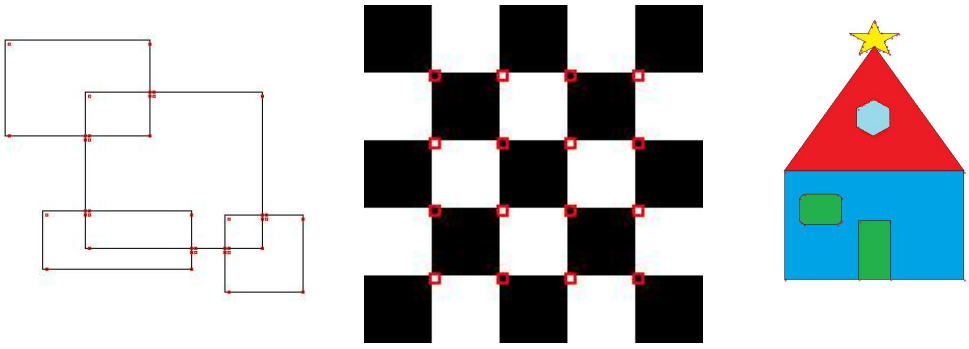
Non-Maximal Suppression functionality is implemented in ***Harris_main(img)*** function.

3.3 Results:

Input Frames:



Output Frames:



4. FINDING MATCHING POINTS USING THE SUM OF SQUARED DIFFERENCE (SSD)

4.1 Algorithm

1. Calculate the lengths of the two lists of corners, I1 and I2, respectively.
2. Create an empty list called corner_map.
3. Loop through each corner in cornerList1.
4. Extract the x and y coordinates of the current corner and create a coordinate list containing the x and y values and the corresponding 10x10 neighbourhood around the corner.
5. Iterate through each corner in cornerList2.
6. Check if the current corner is within the 10x10 neighbourhood of the corner from cornerList1. If it is, compute the SSD between the two corners using their corresponding corner strengths, R1 and R2.
7. If the computed SSD is less than the current minimum SSD, update the minimum SSD and the best matching corner.
8. Append the current corner, its best match, and the corresponding minimum SSD to the corner_map list.
9. After iterating through all corners in cornerList1, return the corner_map list.

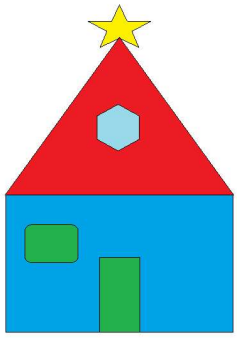
4.2 Implementation

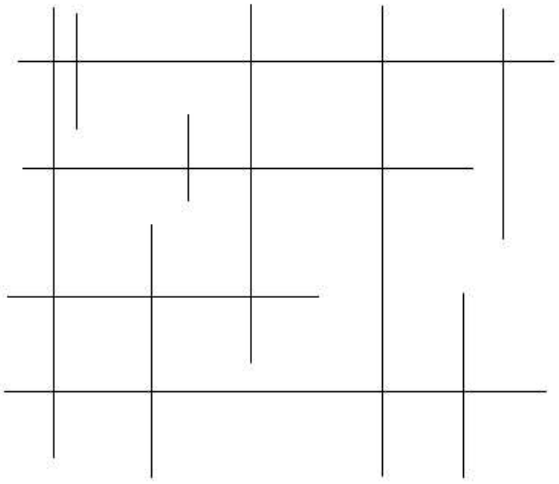
To match the correspondence point between adjacent frames, we have used the method of SUM OF SQUARED DIFFERENCE (SSD),

For this, we have created a patch of 21x21 for each corner, and checked the difference of R values from 1st frame and a patch of 2nd frame. We have mapped the correspondence point with a minimum SSD less than the threshold.

4.3 Results

Input Frames

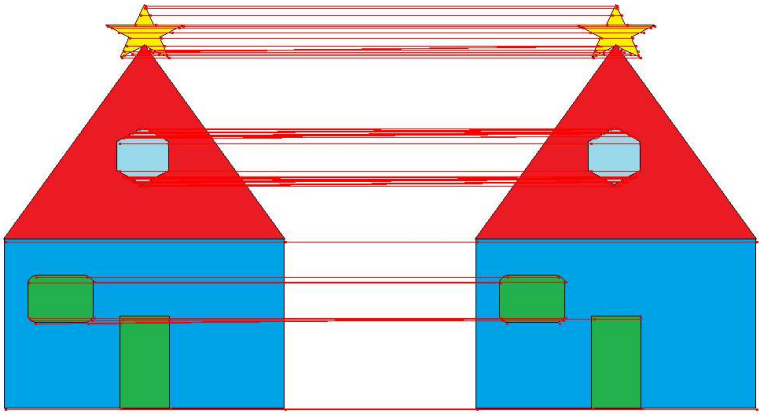


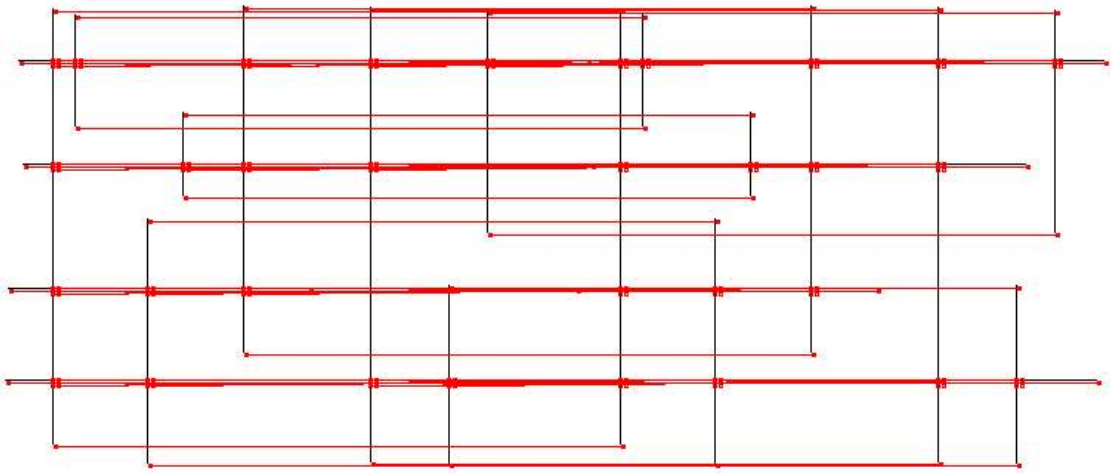


To check the correctness of our algorithm, we have checked our SSD function by giving the same image as image 1 and image 2.

And we got the following results.

Output Frames:





5. IMAGE STITCHING FOR PANORAMA CREATION USING AFFINE MODEL

5.1 Affine Matrix Calculation using DLT

DLT stands for Direct Linear Transform. It involves solving a system of linear equations to find the parameters of a transformation(Affine) matrix that maps points from one coordinate system to another.

Algorithm:

The DLT algorithm takes a set of correspondences between points in two images or 3D scenes as input and outputs a transformation(Affine) matrix that can be used to map points from one coordinate system to the other.

The steps of the DLT algorithm are as follows:

1. Input correspondences between points in two images or 3D scenes.
2. Set up a system of linear equations that relates the coordinates of the points in one coordinate system to the coordinates of the points in the other coordinate system.
3. Solve the system of linear equations using methods such as least-squares or singular value decomposition (SVD) to obtain the transformation matrix.
4. Apply the transformation matrix to the points in one coordinate system to obtain their coordinates in the other coordinate system.

5.2 Image Stitching using Affine Model by Bilinear Interpolation

Image stitching is combining multiple images with overlapping areas to create a larger, panoramic image. One way to achieve image stitching is using the affine model and bilinear interpolation.

Bilinear interpolation is a method for estimating pixel values in an image by interpolating between adjacent pixels. Given four neighbouring pixels (p1, p2, p3, p4)

and their distances to a target point (x, y), the interpolated pixel value (p) can be calculated as follows:

$$p = (1 - u)(1 - v)p1 + u(1 - v)p2 + (1 - u)vp3 + uv*p4$$

5.3 Results

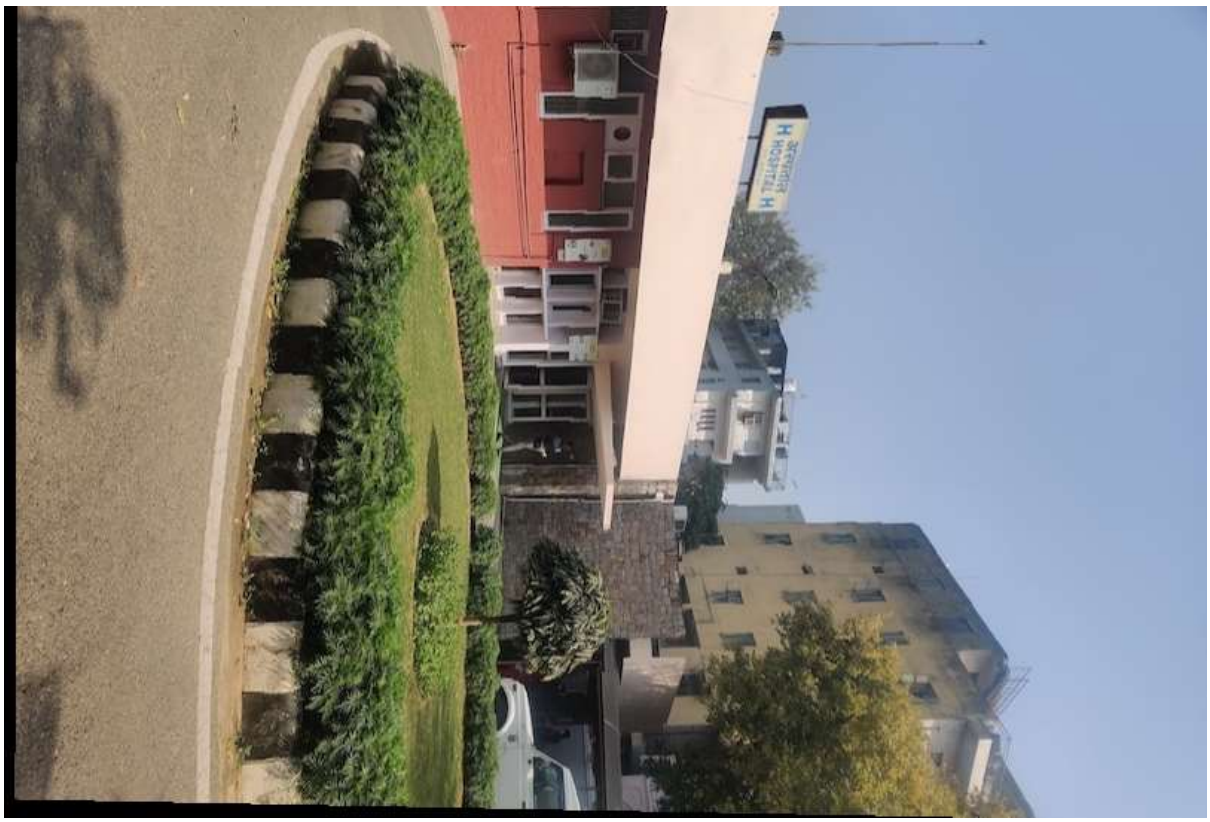
Input Frames:

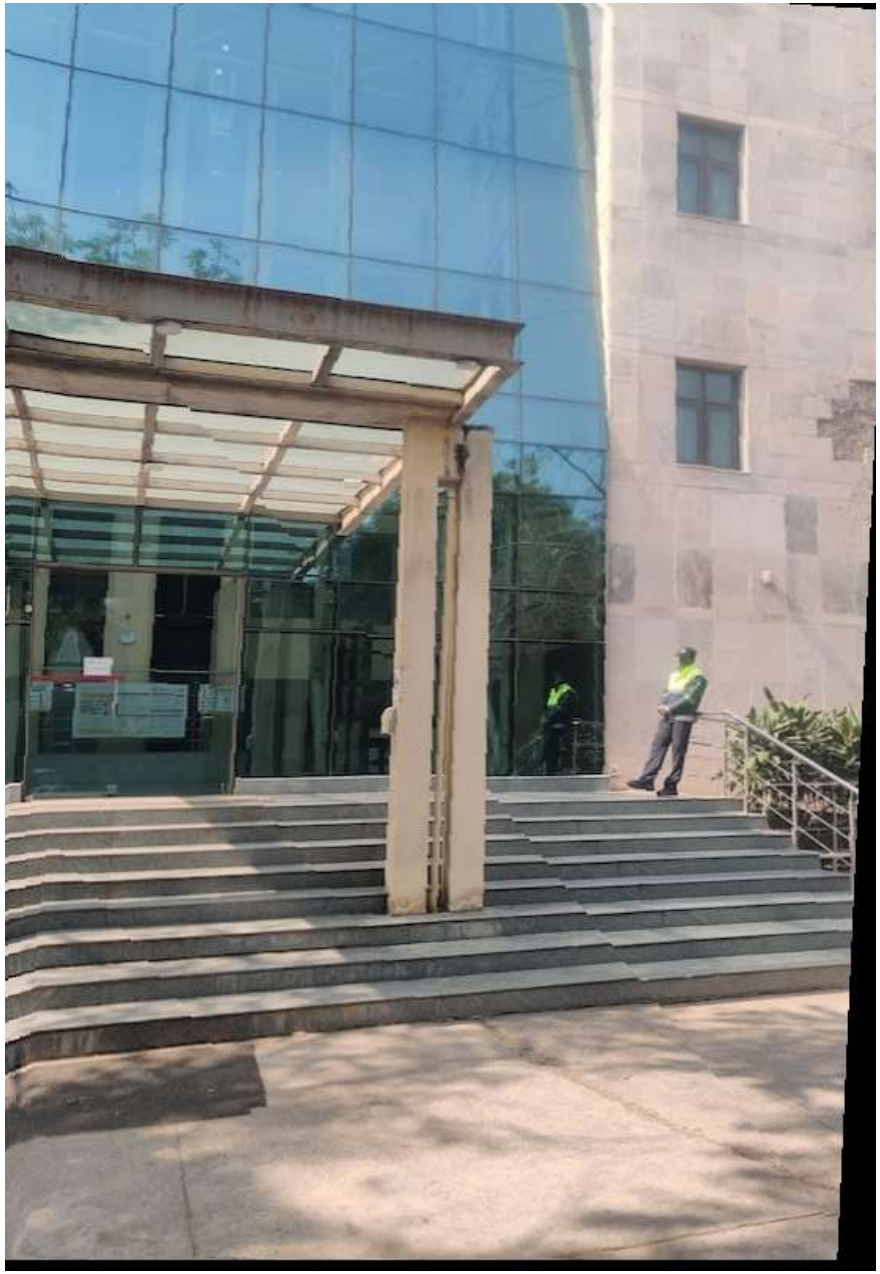
Link:

Output Frames:

Link: Results

2.jpg





6. CONCLUSION

In this work, we presented an implementation of the Harris corner detector. We implemented a generic non-maximum suppression algorithm that allows selecting the prominent features on the image. To improve the speed of the method, it is necessary to implement a faster Gaussian convolution technique or to replace it with a box filter, at the expense of an accuracy loss.
So, we can think corner points depend on the threshold, alpha (smoothing factor) and blur applied to reduce the noise present in the image.

After stitching the image, we noticed that sometimes we got the seam(line), it depends on the quality of the image, ambience and orientation of images so to overcome that, we need to implement the blending function in the particular case.

7. BIBLIOGRAPHY

[1]. Computer Vision: Algorithms and Applications Book by Richard Szeliski

[2]. A COMBINED CORNER AND EDGE DETECTOR Chris Harris & Mike Stephens

http://www.bmva.org/bmvc/1988/avc-88-023.pdf

[3]. https://opencv.org/