COL780
Assignment 1

Avani Jain
2020MT10792

Contents

The value of Y for me is 0.

1 Method for Harris Corner Detection

The solution uses the Harris Corner Detection method very similar to as discussed in class. In addition, it uses the concept of integral images to increase efficiency.

Harris Corner Detector - Compute the Harris corner detector using the following steps:

- Compute the x and y derivatives on the image stored
- Compute the sum of dx^2, dy^2 and $dxdy$ over a window or small area of the image
- Compute the Harris Measure
- Find peaks in the response that are above the threshold "thres", and store the interest point locations

Important points to note :

- Corner detection is implemented not on RGB images but on grayscale to get better results.
- The image is smoothened by applying Gaussian Filter to remove noise.
- The concept of integral images as taught in class is used to increase the efficiency of the code.
- A high value of threshold is kept to get less number of corners detected initially
- NMS technique is used to avoid renundant corner detections, as decsribed below

The threshold for harris measure is set manually for every dataset by observing the obtained number of corners. The alternative to not keep this manual could be implemented directly by using a low threshold and modifying the current code to give a fixed number of corners of all the corners detected that have amongst the highest measures.

(a) Frame 1

(b) Frame 2

Figure 1: Corner detection in dataset 1

```
number of corner pixels:  30
number of final corner pixels:  6
DONE  1
number of corner pixels:  30
number of final corner pixels:  5
DONE  2
number of corner pixels:  31
number of final corner pixels:  6
DONE  3
number of corner pixels:  32
number of final corner pixels:  6
DONE  4
number of corner pixels:  30
number of final corner pixels:  6
DONE  5
number of corner pixels:  30
number of final corner pixels:  5
DONE  6
number of corner pixels:  31
number of final corner pixels:  6
DONE  7
number of corner pixels:  32
number of final corner pixels:  6
DONE  8
number of corner pixels:  30
number of final corner pixels:  6
DONE  9
number of corner pixels:  30
number of final corner pixels:  5
DONE  10
number of corner pixels:  31
number of final corner pixels:  6
DONE  11
number of corner pixels:  32
number of final corner pixels:  6
DONE  12
number of corner pixels:  30
number of final corner pixels:  6
DONE  13
number of corner pixels:  30
number of final corner pixels:  5
DONE  14
```

Figure 2: Dataset 1 terminal output before and after NMS

1.1 NMS to find local maximas

Using simple Harris Corner Detection gave many corners that were in vicinity of each other. This was because we are using a patch of some size to calculate the gradient measure, hence all the values in the vicinity of a Harris corner would also become Harris corner. So, to get around with this problem, the technique of Non Maximal Supression(NMS) to find the local maximums is used.
Algorithm : The model first sorts the initially detected corners in descending order, using the values of the Harris measure, and then iteratively picks any corner that is at least as far as some threshold value from already selected corners.

The detected corners in the first image of the first dataset are as shown in Figure 1. Figure 2 is the terminal output on dataset 1, that prints the number of corner points before NMS and after NMS. The rest of the files are available on the link given in the last section.

2 Matching of corners

Using the constraint that the images have only a little shift, we know that the corners must lie within a range in the other frame.

```
Frames 1 and 2 :
Match 1
[1900. 1681.]
[1900. 1661.]
Match 2
[1932. 2365.]
[1932. 2345.]
Match 3
[1953. 1737.]
[1953. 1717.]
```

```
Frames 2 and 3 :
Match 1
[1900. 1661.]
[1900. 1641.]
Match 2
[1932. 2345.]
[1932. 2324.]
Match 3
[1953. 1717.]
[1953. 1697.]
```

Figure 3: Corner Matching in dataset 1

Algorithm : The corner sets of the two images, that are originally sorted in decreasing order of their threshold values are checked with each other, if the two corners have SSD less than the threshold of 500, then the pair is declared to be a match.
An example of matching is as shown in Figure 3 which is terminal output in the case of dataset 1.

Also, since we are using an affine model it is sufficient to find three matches to know the whole transformation matrix.

3 Stitching of frames

Using 3 pairs of matched points, the matrix for the affine transformation is calculated for each pair of consecutive frames. Then for stitching of frames, first task is to calculate the size of their stitched image, for this I have used the following algorithm :

- Iteratively stitched two images starting from the end using the calculated affine transformations between them.
- For stitching of two images, calculating the size of the stitched image by using the transformation matrix on corners.
- Followed by fixing the starting part of the new image from image 1, and the ending part by image 2. The boundaries can be seen calculated from transformation matrices, as also shown in Figure 4.
- This procedure is repeated for all frames starting from the end.

4 Links of Generated Outputs

Click on this link for the outputs

There are two folders for each dataset - **result**, **final**. The number following them in the name is for the respective dataset as given in the input.
result folder contains the initially detected harris corners after doing part 1 of the assignment, the corner pixels are turned to blue colour and are surrounded by green rectangles.
final folder contains the files obtained in stitching, the numbered files are all obtained in step-wise stitching of the images, and the final panorama image is contained in the file named **"panorama.jpg"** of this folder. The panorama files are also pasted in this document for each dataset.

All the code files as well as the Jupyter notebook, along with the requirements.txt file, have been included in the main folder.

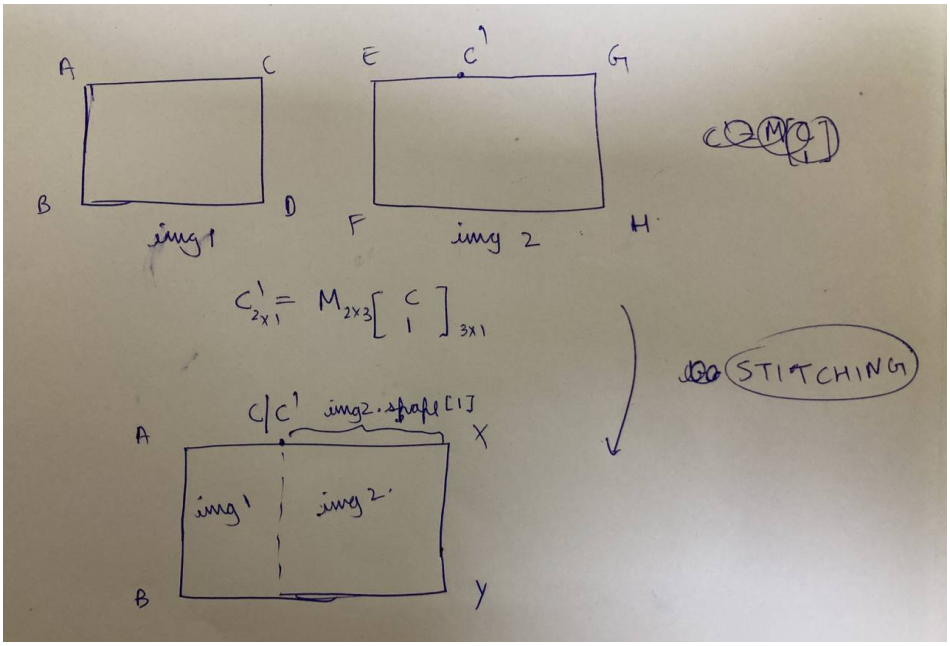

Figure 4: Stitching of images

Figure 5: Output 1

Figure 6: Output 2

Figure 7: Output 3

Figure 8: Output 4

Figure 9: Output 5