Computer Vision : Assignment 2

Name : Sandeep N Menon

Roll No : 14CO140

Harris Corner Detector

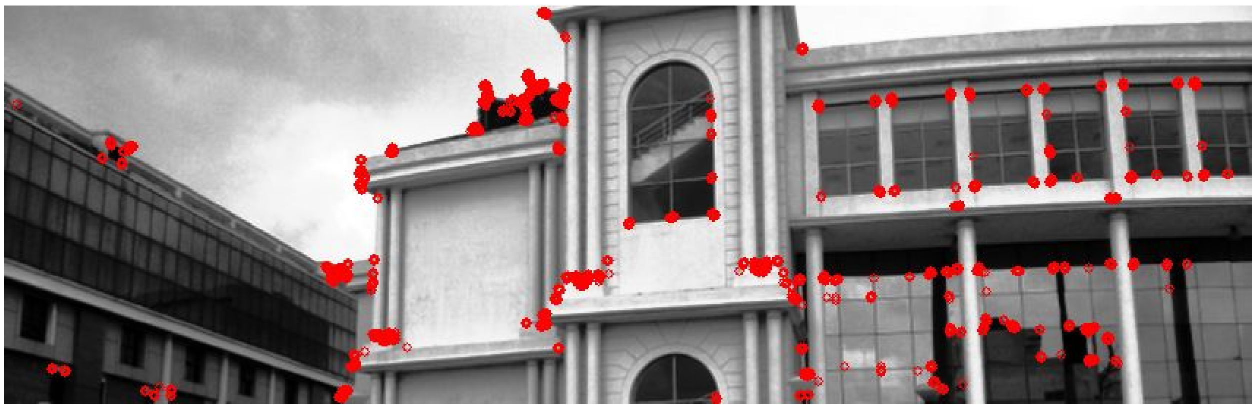
Harris corner detector used in assignment 1 was used with a slight modification. The code in assignment 1 gave all points which passed the threshold to be corners. But this approach has some drawbacks

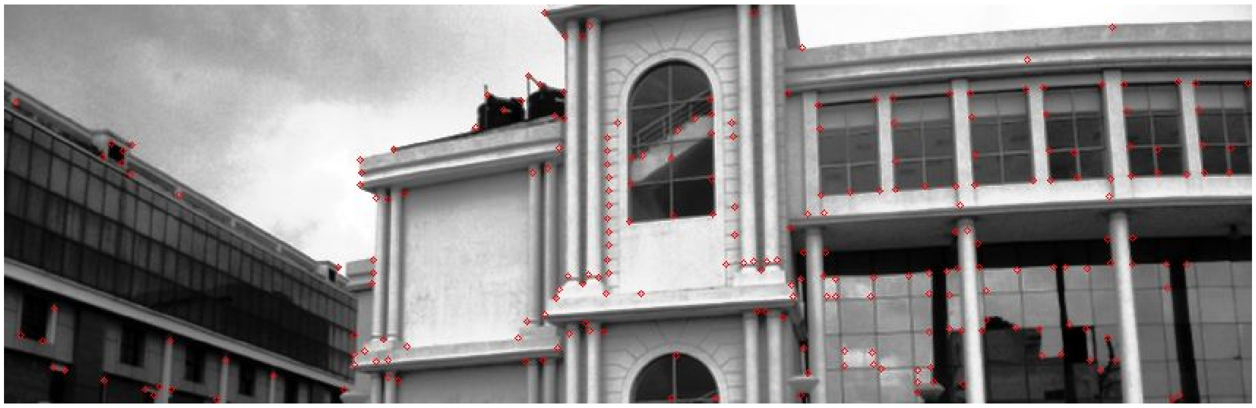
1. Points near an actual corner, has a high R-score thus making all those points getting marked as corner points
2. The point mentioned in point 1 gets even worse while applying blurring techniques to reduce noise. This blurs the corners and gives rise to a myriad of corner points.

The above mentioned problems were addressed as follows

1. If a coordinate crosses the threshold to be a corner, then it is marked as a corner only if its R-score value is higher than its local neighbours.
2. A matrix storing the R-score of all coordinates is created, and for the values that cross the threshold, a check of local maxima is done.

Comparing outputs of previous and modified harris corner detector





The first image is the output from the old harris corner and second picture is after applying the *R-score local maxima* modification.
(“Rbhavan_corner_marked_image.png” and **“rbhavan_newcorners.png”)**

Images Used for the experiment

The images can be found in files “**Limage.png”** and “**Rimage.png”.**

Steps Before Projective Transformation

1. Noise Reduction : Gaussian Blur of kernel size 3x3 was applied to both the images to reduce noise and avoid unwanted corner detections
2. Increase contrast : Contrast of the images was increased using Histogram Equalisation to increase the intensity level differences at edges and corners.
3. Harris corner detector was applied to both the images to find and mark the corner points on both the images.
    a. Output image files are “**Limage_corners**.**png”** and “**Rimage_corners.png”**
4. Observing the output from the harris corner, the common corners were manually marked and noted down here **“cornerlist.txt”**.

Projective Transformation and Stitching

Steps Involved

1. Finding Homography matrix

The relation between projection coordinates between two systems is calculated using a homography matrix. For a projective transformation there are 8 degrees of freedom. Hence we require at least 4 sets of points from both the images so that we have 8 equations to calculate the 8 unknowns. But, in practice, we can use more than 8 points to get a better approximation of the homography matrix.

The function cv2.findHomography(), calculates the homography matrices from the two sets of points passed to it.

The steps for calculating homography matrix is given below.

a. Identify the two sets of points that the homography matrix should projective transforms
b. Express the homography matrix in terms of the matrix multiplication of the points in matrix form and their inverses.
c. Compute “Singular Value Decomposition” of the above matrix equation to obtain the homography matrix.

For this experiment, and the corner points mentioned in **cornerlist.txt ,** we obtained a homography matrix of

```
[  1.24961539e+00,   2.01578464e-02,  -5.74777977e+02],
[  2.38311603e-01,   1.19052608e+00,  -1.37854402e+02],
[  3.14141502e-04,   1.23588045e-05,   1.00000000e+00]
```

2. Defining canvas dimensions for the two images of the final stitch
3. Perspective transform on one image dimensions using the homography matrix
    a. In this experiment, for an image of size **1280x720**, the dimensions before transformation is

```
[[[    0.     0.]]

 [[    0.  1280.]]

 [[  720.  1280.]]

 [[  720.     0.]]],
```

b. And after transformation is

```
[[[ -574.7779541   -137.85440063]]

 [[ -540.42675781  1364.43457031]]

 [[  282.40484619  1254.10778809]]

 [[  265.00564575    27.50811386]]]
```

4. Warping one image into the canvas plane. This includes
    a. Translating the image by its minimum x-coordinate and y-coordinate
    b. **cv2.warpPerspective(img1, transform_array.dot(homography_matrix), (x_max-x_min, y_max-y_min))** .
5. Placing the other image into the canvas and thus stitching the two images

OUTPUT

The output images are given as follows

1. **Limage_commoncorners.png** and **Rimage_commoncorners.png** are the common corner points on the original images,
2. Final stitched image is **result.png.**