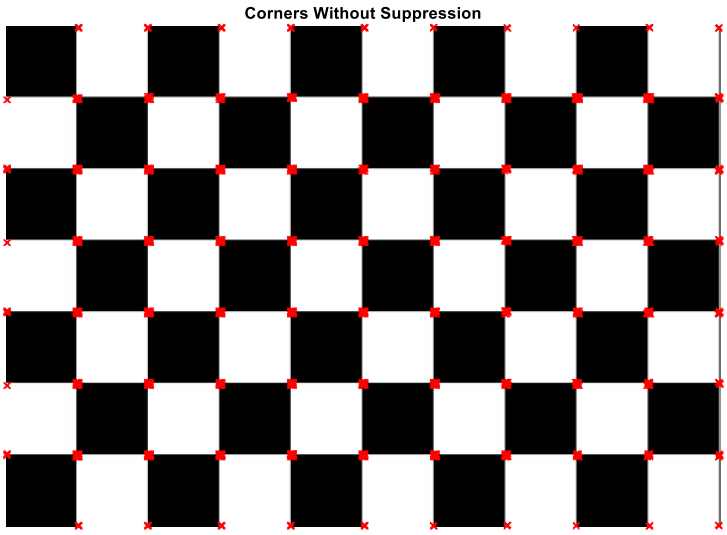
CSCI 5561 – HW3 – Rohith Nallamaddi

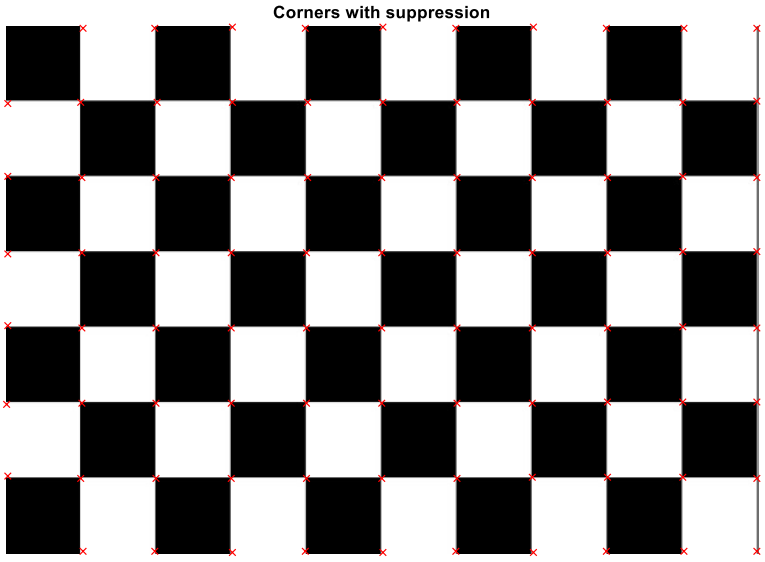
Part One

1) **Results:**

Corner.m - the below 2 figures show outputs with and without suppression

Funtion definition:
function [rows,cols] = corner(I1,thresh,supp_window)
I1 : Input Image
thresh : threshold on the min eigen value. Empirically thresh = 0.2 gave good results
supp_window : Non maxima suppression window size = 15





cornerNoble.m - the below 2 figures show outputs with and without suppression

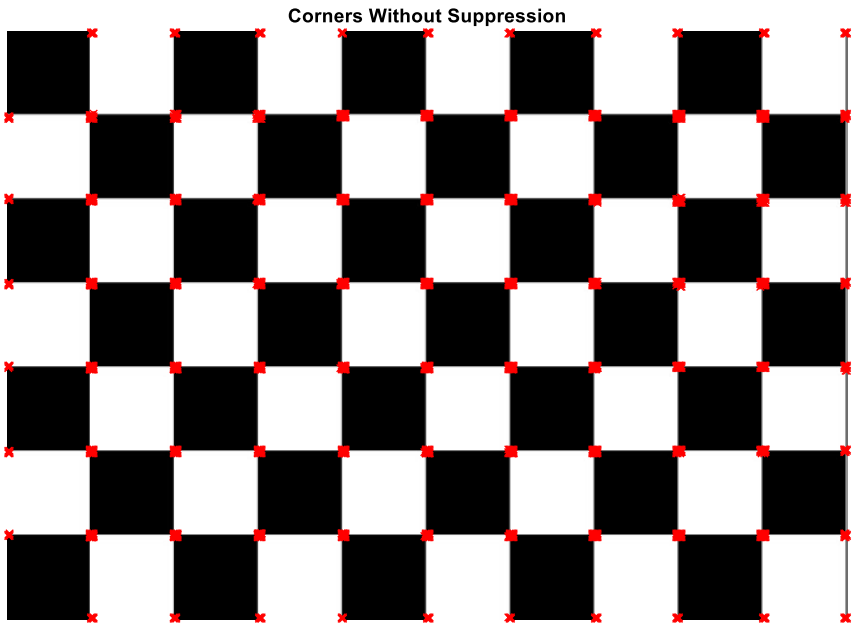
Funtion definition:

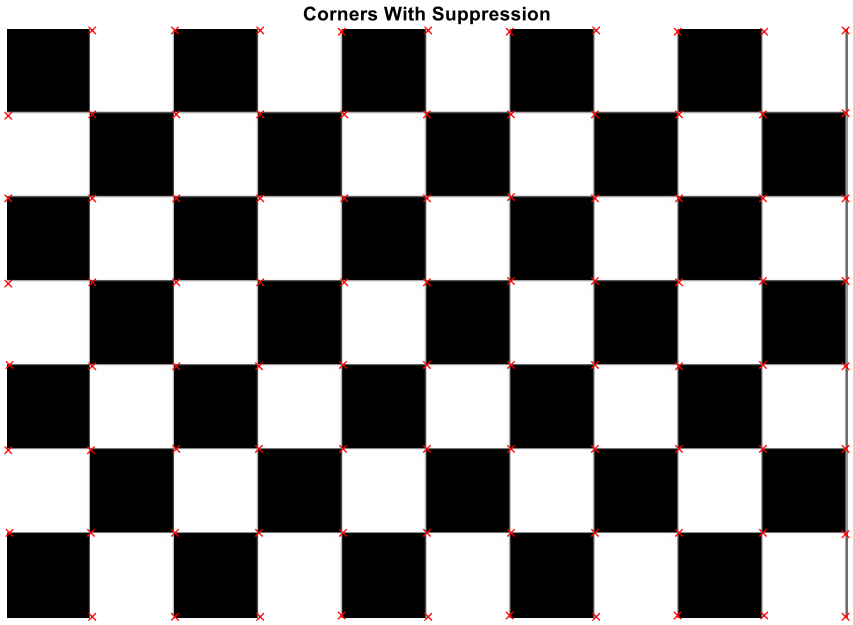
function [rows,cols] = cornerNoble(I1,thresh,supp_window)

I1 : Input Image

thresh : threshold on the Mc value. Empirically thresh = 0.2 gave good results

supp_window : Non maxima suppression window size = 15

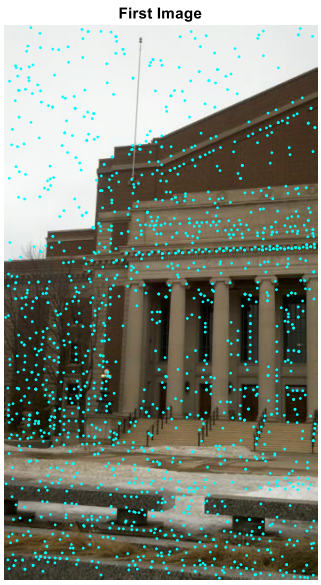


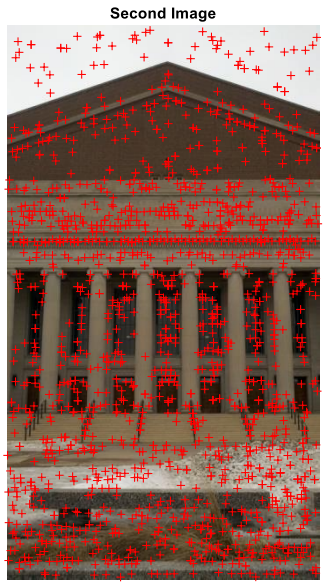


Explanation:
To find the corners we use Harris corner detector which uses the eigen values to detect corners. Both eigen values are larger only for the case of a corner. Only one eigen value is large for a edge and both are small for planar region. Using this concept, we found corners. In cornerNoble, a cornerness measure was defined which is high only when both eigen values are large.

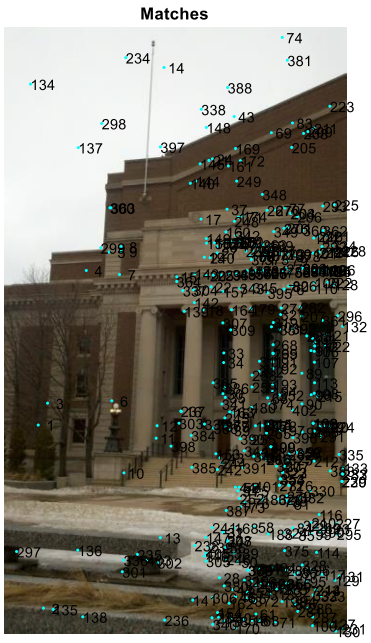
Part Two

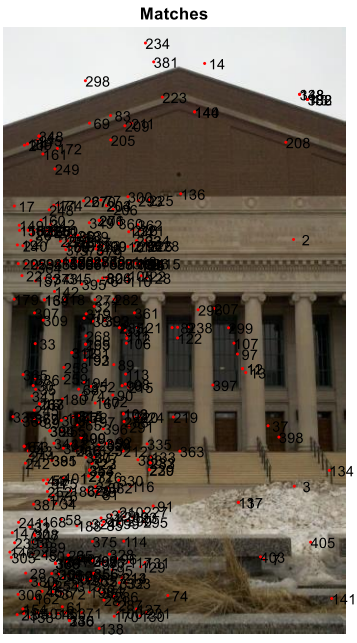
2. SIFT Feature Detection and Matching. Pease run the code q2_sift.m to generate the results.
Detected features in both images:





Matches labelled with numbers





3.1 Homography Estimation using Least squares

```
function H = compute_homography_LS(P1,P2)
```
H : Homography matrix

P1, P2: Nx3 inputs

3.2 Homography Estimation using Least squares

```
function H = compute_homography_RANSAC(P1,P2)
```
H : Homography matrix

P1, P2: Nx3 inputs

4. Image Stitching

Function definition:

```
function [IM] = stitch_images(IM_left,IM_right,to_left)
```
IM_left : left image

IM_right: right image

To_left: binary flag to determine the direction of stitching

Please pass the arguments in order. For example, IM_left should be towards left of IM_right in reality as well for image stitching to work properly.

Explanation:

The two images are related by a homography because it just a pure rotation of camera that is yielding us the second image. Pure camera rotation is a homography rotation as the camera translation vector remains the same for both frames and we just need to compute the r1, r2, r3 the camera rotation vectors along three axes. This can also be explained by the fact that for pure rotation, straight lines in image1 remain as straight lines in image2 as well.

4.1 Blending

I used alpha blending on the overlap region of stitched images to smoothen images.

I also tried blending using laplace and Gaussian pyramid but the outputs were not so promising. I have included the script **stitch_blend_images.m** which does this.

Results:

The figure below shows mall1 projected into mall2 frame.

The figure below shows mall4 projected into mall5 frame

The figure below shows mall4 projected into mall3 frame

The figure below shows mall1 projected into mall2 frame

Stitched Image

5. Panorama

Results:

panorama output

Part 3

Function Definition:

```
function [IM_AR, Cam_R, Cam_t] = AR_Cube(IM, Ang, t)
```

IM: Input image
Ang: angle of rotation in degree
t: row vector specifying the translation wrt origin

Explanation:

I first estimated the camera matrix using the homography. We could estimate the camer matrix from single image because the image lies on the plane and as we set Z = 0, the effect of r3 in camera matrix is nullified. Therefore we can compute the camera matrix from homography. Later r3 is found by cross(r1,r2).

To project cube on to the images, I first defined cube world coordinates and used the camera projection matrix to project on to the paper. I made sure that only visible faces are displayed by checking the sign with camera centre.

Results:

Table1

Cube on the paper displaying visible faces

Table2

Cube on the paper displaying visible faces

Table 3

Cube on the paper displaying visible faces

Table 4

Cube on the paper displaying visible faces

Table 5:

Cube on the paper displaying visible faces