CS-867 Spring 2021

Computer Vision

Assignment # 01

Harris Corner Detector & Its robustness to Rotation & Scaling

Instructor: Dr. Shahzor Ahmed

Submitted by: Monazza Qadeer Khan

Id: 206154

Dated: 1st May, 2021

Harris Keypoint Detector and its Robustness to Rotation and Scaling

PART- A

Implement the Harris keypoint detector as discussed in the lectures. You do not need to implement a descriptor. Apply it to the two images provided and threshold the cornerness response so that about a few hundred keypoints are returned. Submit a result showing the (top-scoring) detected Harris keypoints superimposed on the original image (e.g., you can place a cross or a circle at the location of the keypoint; do this on the grayscale version to facilitate visualization). Report the threshold that you choose. Describe which objects or regions in the image seem to generate large numbers of Harris keypoints, and why. You may **not** use the built-in library function for Harris detection, but must implement it on your own.

Programming platform: MATLAB

The following built-in functions are used for computations:

- im1 = imread('famous_five.png') for image reading
- im = rgb2gray(image) for converting color image to grayscale
- ix = conv2(double(im),dx,'same') for horizontal derivative
- iy = conv2(double(im),dy,'same') for vertical derivative
- PARAMETERS FOR GAUSSIAN FILTER
  - sigma = 3
  - order = 16 * 16
- Ix2 = conv2(double(ix.^2),G,'same') for Ix^2
- Iy2 = conv2(double(iy.^2),G,'same') for Iy^2
- Ixy = conv2(double(ix.*iy),G,'same') for Ix*Iy
- Threshold =4000
- maximum_point = ordfilt2(r, order^2,ones(order)) filter for non-max suppression
- Order of NMS filter =32 * 32
- harris_corners = (r==maximum_point) & (r>threshold) for finding Corners

Note: I resized the given images to a standard size i.e. 640 by 480 in a photo editor before using them in MATLAB, just to avoid any ambiguity and to focus on the main task.

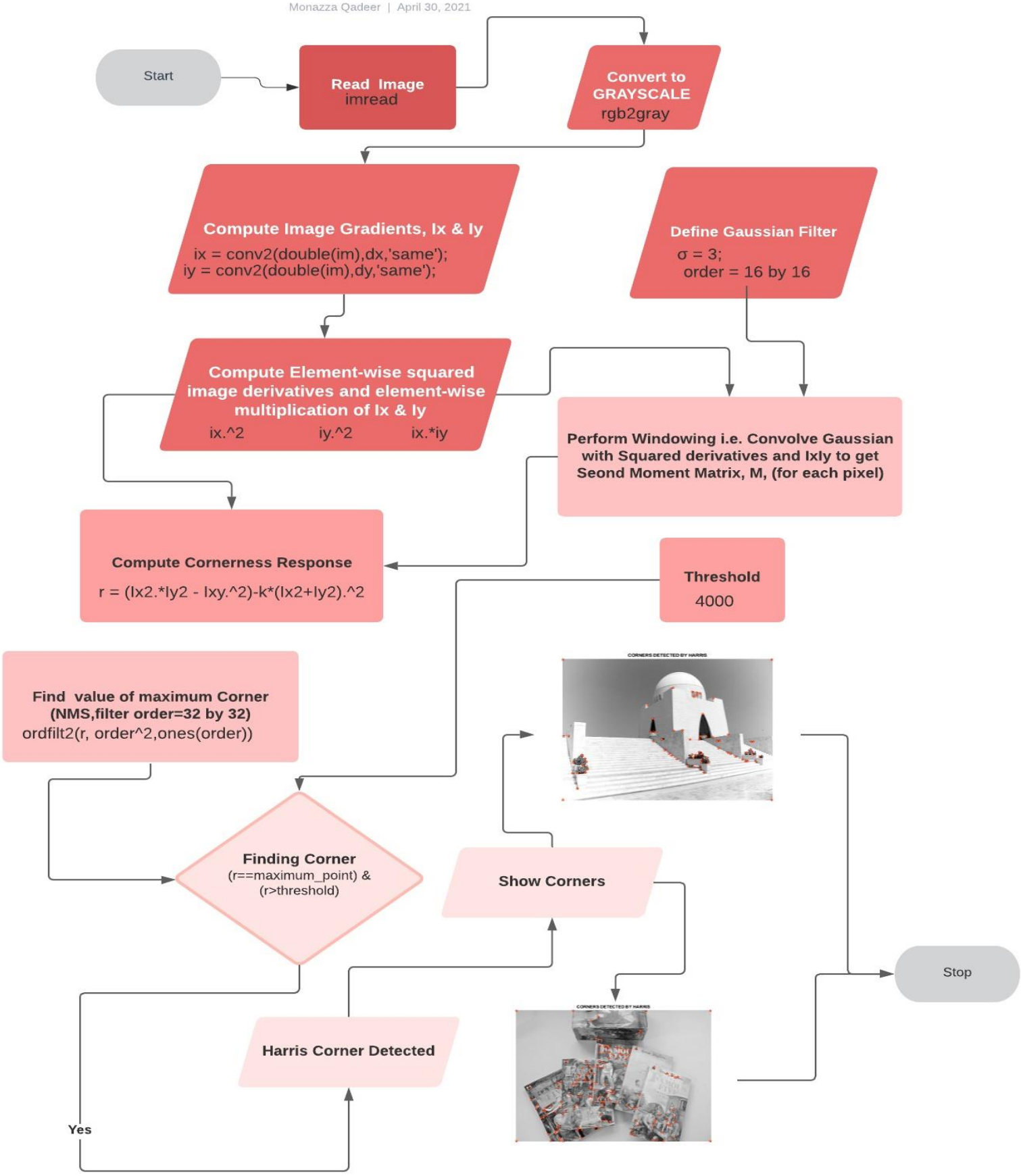
Code

```
% CS-867 COMPUTER VISION
% ASSIGNMENT-1, PART-A
% HARRIS KEYPOINT DETECTOR

% Read given images
im1 = imread('famous_five.png');
im2 = imread('mausoleum.jpg');
harris(im1);
harris(im2);

function harris(image)
% Displaying given image
figure;
imshow(image);
title('GIVEN IMAGE');

% Converting given RGB image to GRAYSCALE
im = rgb2gray(image);
% Displaying GRAYSCALE image
figure;
imshow(im);
title('GRAYSCALE IMAGE');
```

```
% Finding Horizontal & Vertical Gradient of GRAYSCALE image
[dx,dy]=meshgrid(-1:1, -1:1);
ix = conv2(double(im),dx,'same');
% Displaying Horizontal gradient/ Vertical Edge
figure; imshow(ix);
title('HORIZONTAL GRADIENT, Ix');
iy = conv2(double(im),dy,'same');
% Displaying Vertical gradient/ Horizontal Edge
figure; imshow(iy); title('VERTICAL GRADIENT, Iy');

% PARAMETERS FOR GAUSSIAN FILTER
sigma = 3;
radius=1;
order = (2*radius+1)^2;

% DEFINING GAUSSIAN FILTER
len = max(1,fix(6*sigma));
p=len; q=len;
[u1,u2]=meshgrid(-(p-1)/2:(p-1)/2, -(q-1)/2: (q-2)/2);
ug = exp(-(u1.^2+u2.^2)/(2*sigma^2));
[w,z] = size(ug);
sum = 0;
for i=1:w
    for j=1:z
        sum = sum+ug(i,j);
    end
end
G = ug ./sum;

% COMPUTING ELEMENTS OF SECOND MOMENT MATRIX, M
Ix2 = conv2(double(ix.^2),G,'same');
Iy2 = conv2(double(iy.^2),G,'same');
Ixy = conv2(double(ix.*iy),G,'same');

% CORNERNESS MEASURE
r = (Ix2.*Iy2 - Ixy.^2)./(Ix2+Iy2 + eps);

% Value of Threshold set empirically
threshold = 4000;
% FINDING MAX POINT FOR NON-MAX SUPPRESSSION
maximum_point = ordfilt2(r, order^2,ones(order));

% FINDING CORNERS
harris_corners = (r==maximum_point) & (r>threshold);
[R,C]=find(harris_corners);
figure,imshow(im),hold on,
plot(C,R,'yp','MarkerFaceColor','y'),
title('CORNERS DETECTED BY HARRIS');
end
```

HARRIS CORNER DETECTOR

Monazza Qadeer | April 30, 2021

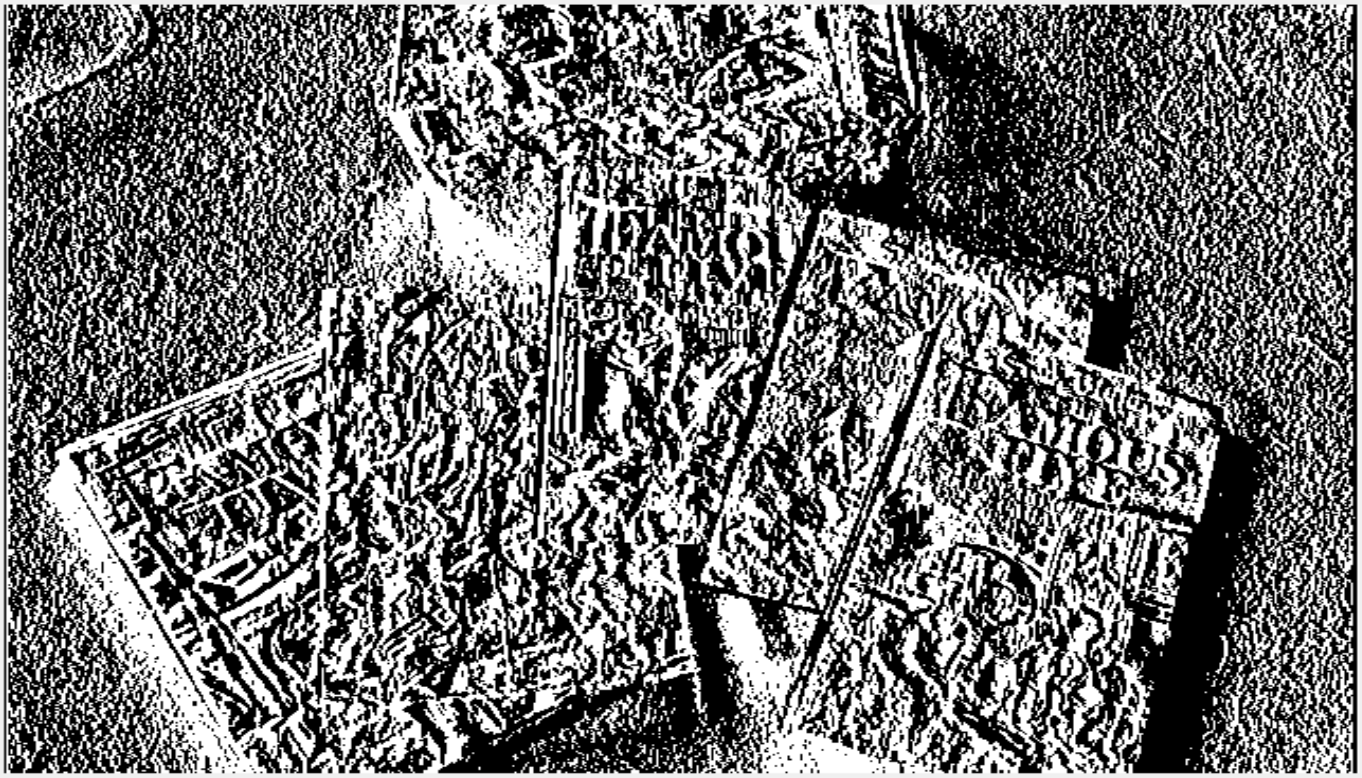


Results

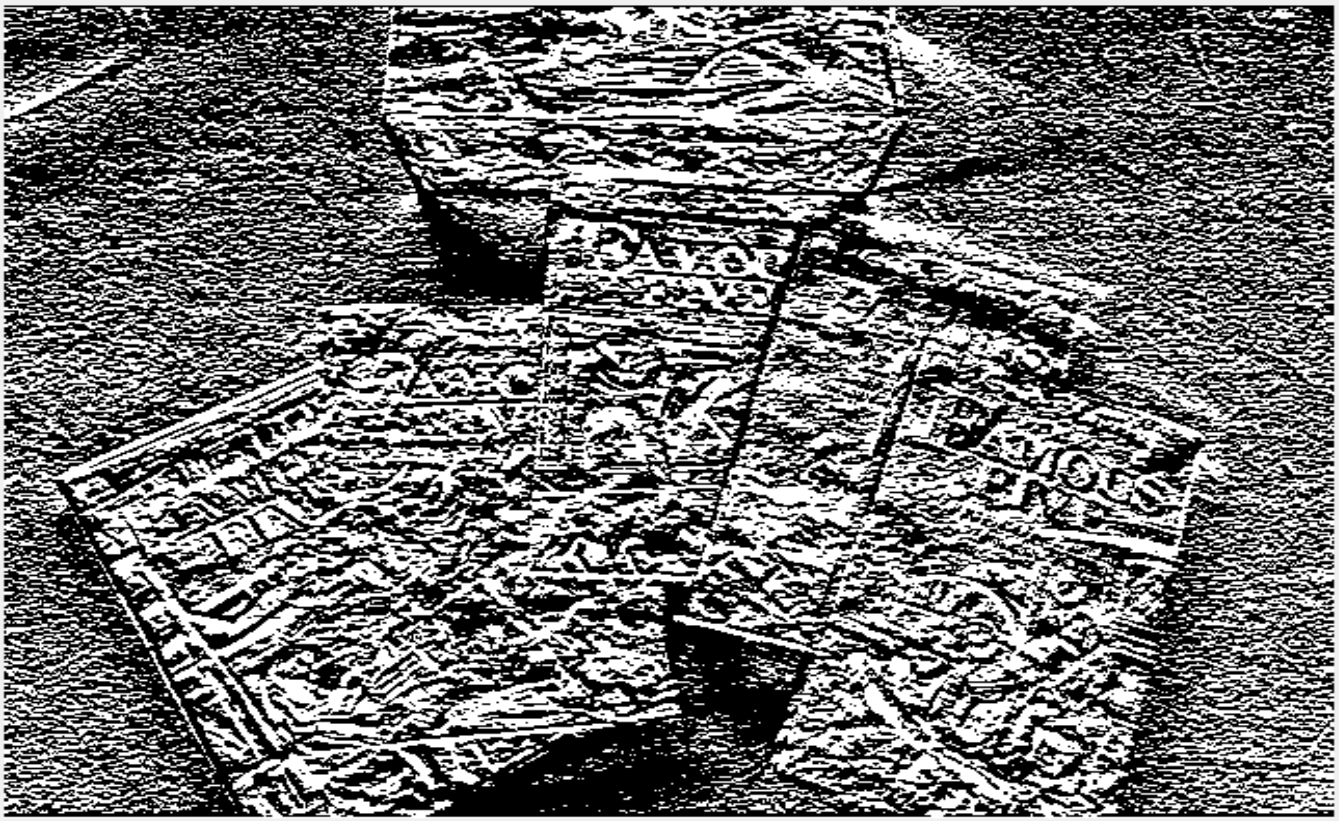
GIVEN IMAGE

GRAYSCALE IMAGE

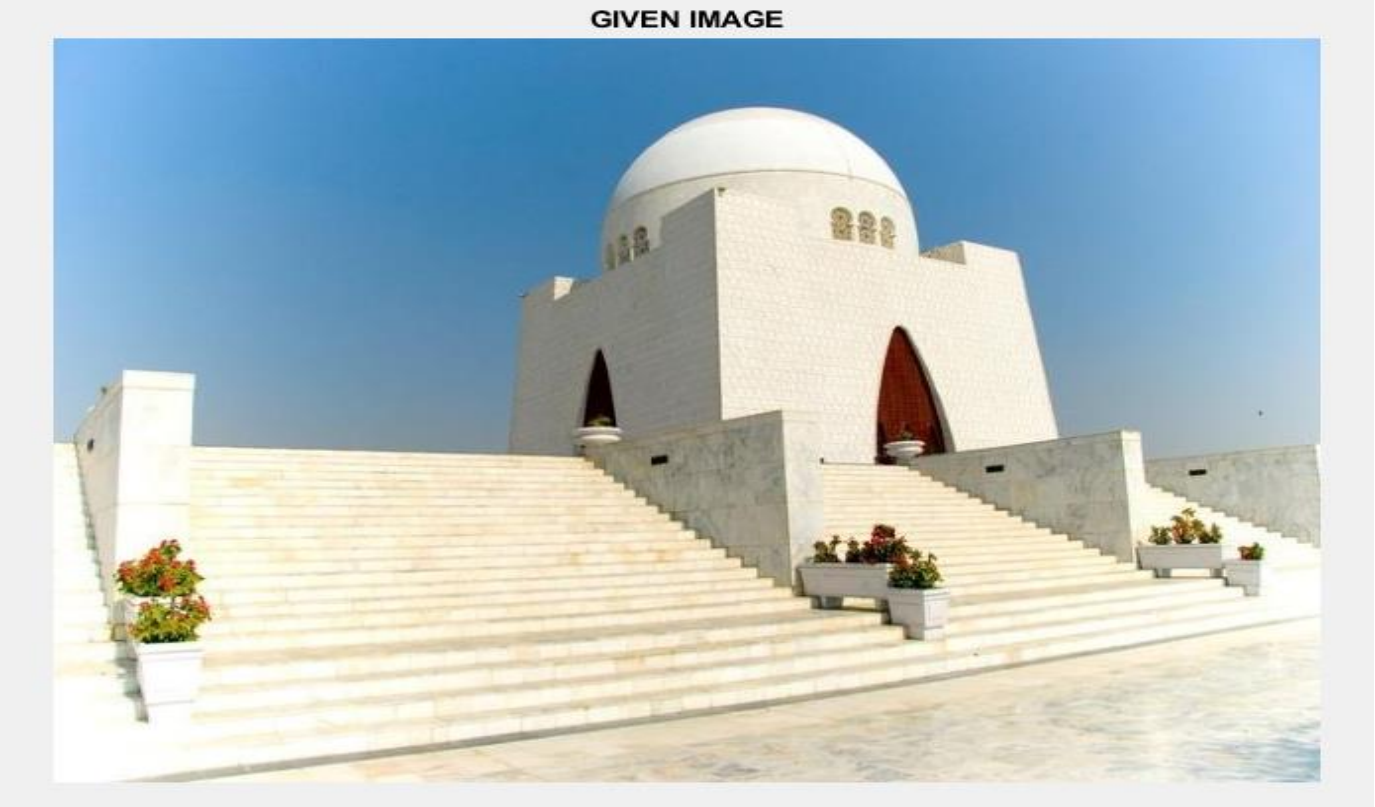
HORIZONTAL GRADIENT, Ix



VERTICAL GRADIENT, Iy

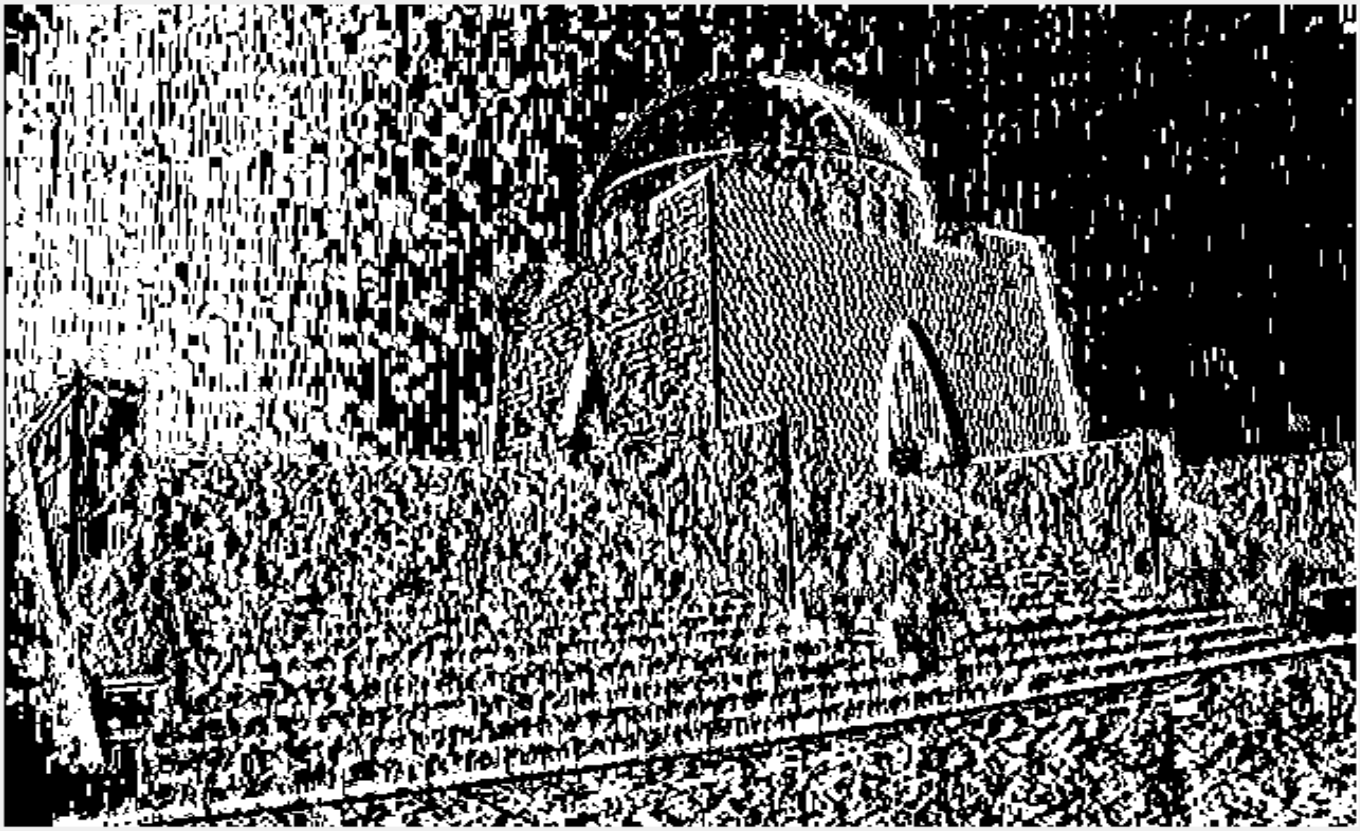


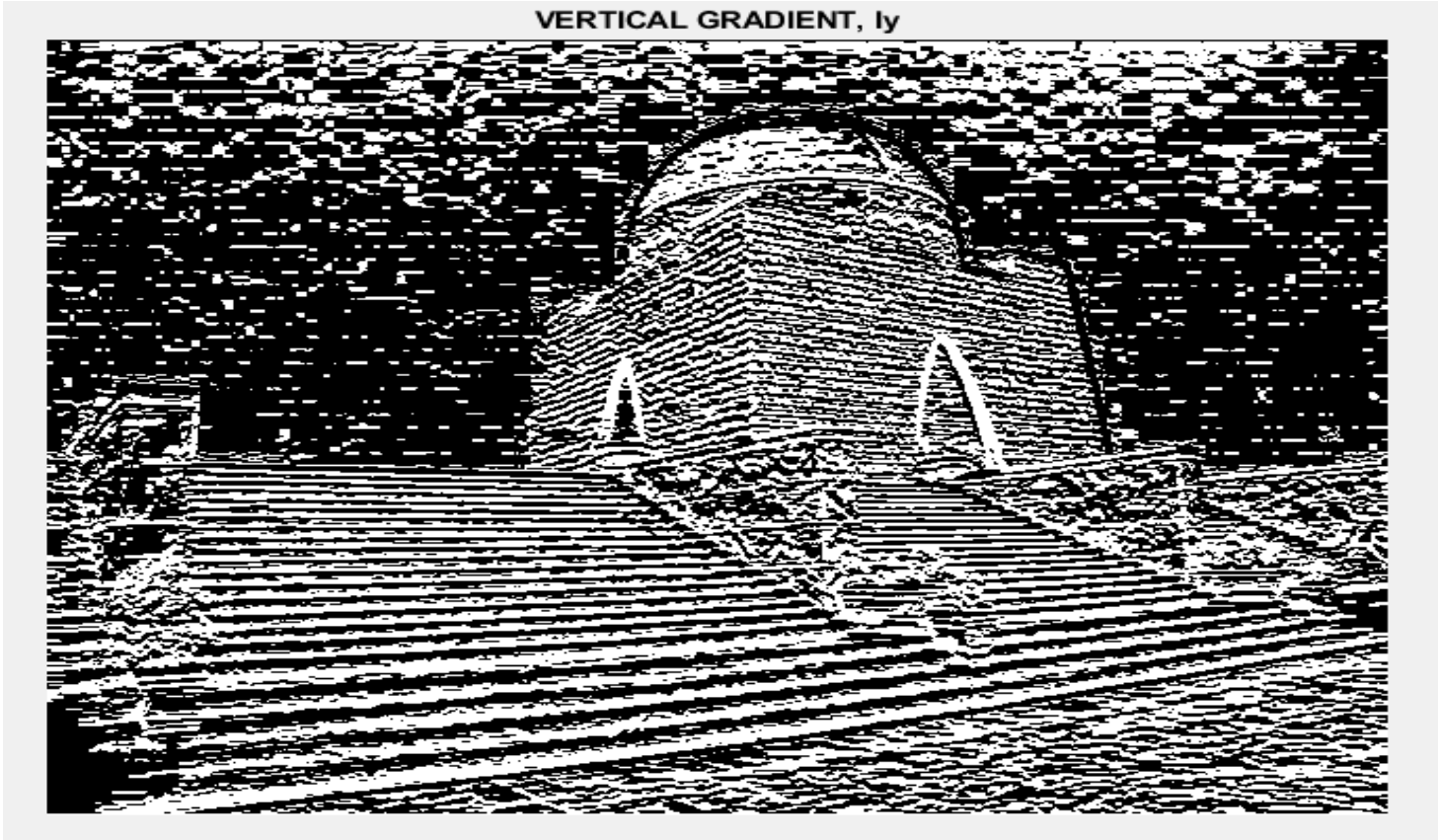
Yellow stars in the following image shows detected corner poinst.



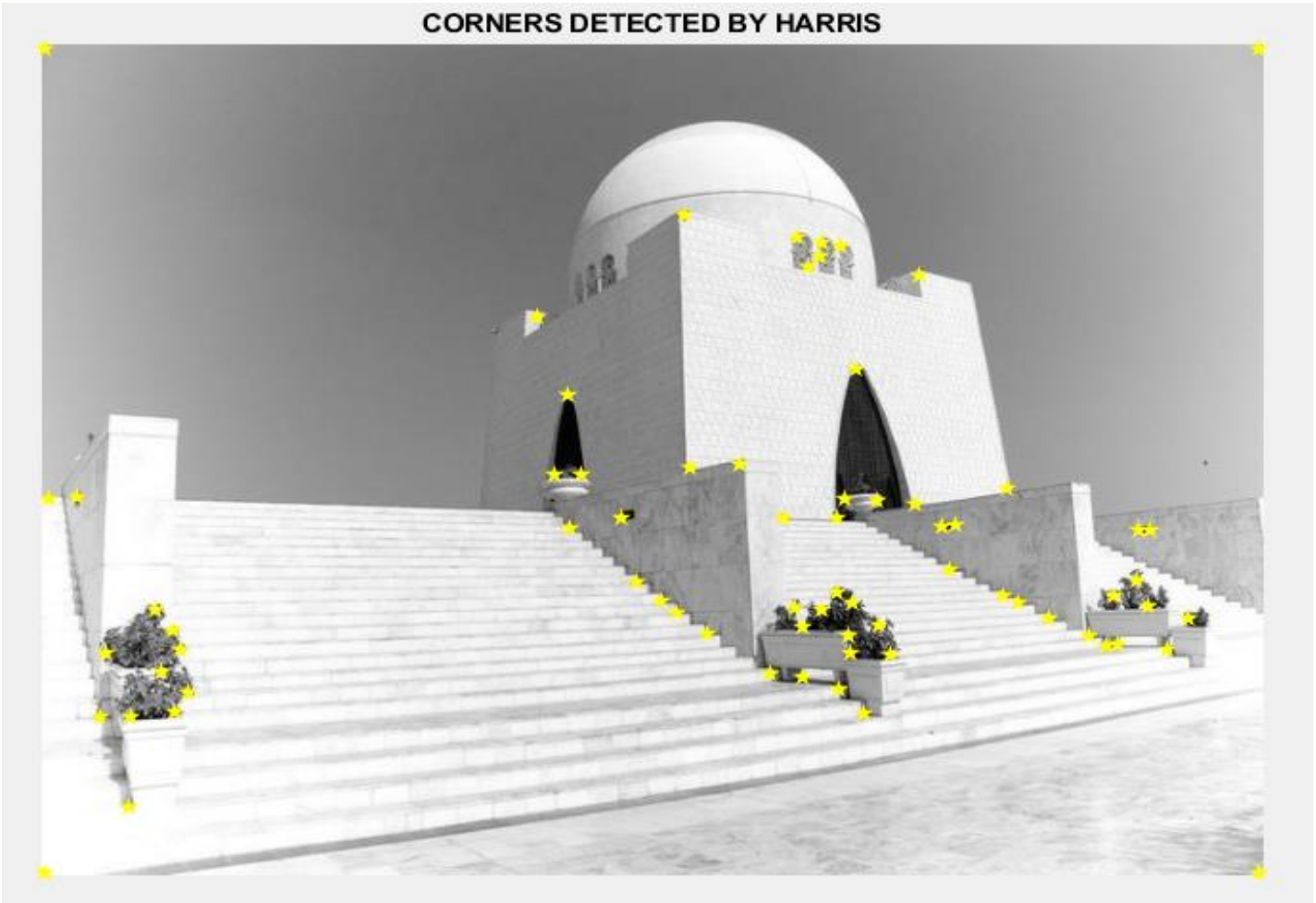
GRAYSCALE IMAGE

HORIZONTAL GRADIENT, Ix





Yellow stars in the following image shows detected corner points.



Conclusion

The yellow stars in the grayscale images depicts the corner points. A medium value of threshold (4000) is set, not too high to miss out on potential keypoints and not too low to get redundant keypoints and also to avoid any ambiguity i.e. to get clear, visible and well separated keypoints for better visualization.

Famous five, the image

This image has good corners as there is a lot of color change even within a single book cover. So, a set of books gives rise to good potential corners such as:

- Name of author at the top of book cover, “Enid Blyton”
- Title of the book series (Famous Five), written in large font
- Paper embossing giving depth to the written words, it creates good corners
- Kids playing around doing different kinds of activities, giving detail to the image, creates good corner points

Mausoleum, the image

As the image contains good and sharp corners so it can be easily found in the following locations:

- **flower pots**, as there is a sharp rate of change in intensity of pixels in every direction with green leaves and red flowers making a good combination for a corner point
- the finishing points or **ends of stairs** also makes a good corner point
- **the sharp corners of mausoleum itself**
- **the small windows** in the dome or vault of mausoleum

PART- B

Rotate the image in increments of 15°, from 0° to 360°. You are allowed use of library function(s) for rotation. For each rotated image, compute Harris keypoints using the same settings that you chose in part (a). Then, compute the repeatability using the following procedure:

- Set number of feature matches M to be 0
- For each Harris keypoint at $[x\ y]$ in the original image:
  - Predict the position $\mathbf{x_r} = [x_r, y_r]$ where the keypoint should ideally appear in the rotated image (by applying a rotation transformation on $[x\ y]$).
  - Search for a nearby Harris keypoint which is detected in the rotated image with coordinates $\mathbf{x_0} = [x_0, y_0]$ satisfying $\|\mathbf{x_0} - \mathbf{x_r}\| \leq T$, where T is some suitably chosen threshold.
  - If such an $\mathbf{x_0} = [x_0, y_0]$ is found, increment number of feature matches M by 1. This matching point $\mathbf{x_0}$ in the rotated image should now not be considered for matching in subsequent iterations.
- Compute repeatability as M/N, where M is the number of feature matches and N is the number of Harris keypoints in the original image.

Plot repeatability against rotation angle and comment on the Harris keypoint detector's robustness against in-plane rotation.

The following built-in functions are used for computations:

- Rimage1 = imrotate(Oimage1,angle) for image rotation
- RotMatrix = [cosd(rot_ang) -sind(rot_ang);sind(rot_ang) cosd(rot_ang)], this is Transformation Matrix for rotation of Corner point
- RotatedP = RotMatrix*((Ocpoints(i))'-CenterO)+CenterR for finding rotated keypoint
- thresh = 100, value of threshold as a reference for finding corner whose Euclidean distance must be less than the set threshold
- Euc_D = sqrt(sum((Rcpoints(i)-RotatedP).^2)) for finding Euclidean distance between predicted keypoint location and keypoint in rotated image

Code

```
1
2       % CS-867 COMPUTER VISION
3       % ASSIGNMENT-1,PART-B
4       % ROBUSTNESS OF HARRIS KEYPOINT DETECTOR TO ROTATION
5
6 -     Oimage1 = imread('famous_five.png');
7 -     Oimage2 = imread('mausoleum.jpg');
8 -     rot_eff(Oimage1);
9 -     rot_eff(Oimage2);
10
11      function rot_eff(Oimage1)
12 -    [Orows,Ocols] = harris(Oimage1);
13 -     Ocpoints = [Orows,Ocols];
14
```

```
% ROTATING IMAGE IN INCREMENTS OF 15 Deg ALL THE WAY FROM 0 Deg to 360 Deg
for angle=0:15:360
    Rimage1 = imrotate(Oimage1,angle); % ROTATE THE IMAGE
    [Rrows,Rcols] = harris(Rimage1);
    Rcpoints = [Rrows,Rcols];
    threshold = 4000;
    Ocpointssize = length(Ocpoints);
    N=length(Ocpoints);% COUNT OF HARRIS CORNERS IN ORIGINAL IMAGE
    rot_ang=15*pi/180; % ROTATION ANGLE FOR IMAGE

    % TRANSFORMATION MATRIX FOR ROTATION OF KEYPOINT
    RotMatrix = [cosd(rot_ang) -sind(rot_ang);sind(rot_ang) cosd(rot_ang)]

    % RETURNS THE CENTER OF ORIGINAL IMAGE(GRAYSCALE)
    CenterO = (size(rgb2gray(Oimage1))/2)';

    % RETURNS THE CENTER OF ROTATED IMAGE(GRAYSCALE)
    CenterR = (size(rgb2gray(Rimage1))/2)';
    M=0; % COUNT OF MATCHED CORNERS
```

```
34
35          % PREDICTING IDEAL POSITION OF KEYPOINT
36 -      for i=1:1:Ocpointssize
37          % KEYPOINT ROTATED BY TRANSFORMATION MATRIX
38 -        RotatedP = RotMatrix*((Ocpoints(i))'-CenterO)+CenterR;
39          % EUCLIDEAN DISTANCE BTW TRANSFORMED & ROTATED IMAGE KEYPOINT
40 -        Euc_D = sqrt(sum((Rcpoints(i)-RotatedP).^2));
41 -        if(Euc_D<threshold) % IF DISTANCE LESS THAN THRESHOLD, THEN
42 -        M=M+1; % INCREMENT IN COUNT OF MATCHED KEYPOINTS
43 -        end
44 -      end
45 -      rep=M/N; %REPEATABILITY, M= MATCHED KEYPOINTS & N= ORIGINAL KEYPOINTS
46 -      figure,imshow(Rimage1),hold on,
47 -           plot(Rcols,Rrows,'yp','MarkerFaceColor','y'),
48 -           title(['Rotated image with ',num2str(angle),
49                ' degree where M = ',num2str(M),' , N = ',num2str(N),
50                ' and Repeatability M/N = ',num2str(rep)]);
51
52 -      end
53 -      end
54
55
56        % HARRIS KEYPOINT DETECTOR AS IN PART-A
57      function [R,C] = harris(image)
58 -      im = rgb2gray(image);
59 -      [dx,dy]=meshgrid(-1:1, -1:1);
60 -      ix = conv2(double(im),dx,'same');
61 -      iy = conv2(double(im),dy,'same');
62        % PARAMETERS FOR GAUSSIAN FILTER
63 -      sigma = 3;
64 -      radius=1;
65 -      order = (2*radius+1)^2;
66        % DEFINING GAUSSIAN FILTER
67 -      len = max(1,fix(6*sigma));
68 -      p=len; q=len;
69 -      [u1,u2]=meshgrid(-(p-1)/2:(p-1)/2, -(q-1)/2: (q-2)/2);
70 -      ug = exp(-(u1.^2+u2.^2)/(2*sigma^2));
71 -      [w,z] = size(ug);
72 -      sum = 0;
73 -      for i=1:w
74 -          for j=1:z
75 -              sum = sum+ug(i,j);
76 -          end
77 -      end
78 -      G = ug ./sum;
```

```
79
80      %COMPUTING ELEMENTS OF SECOND MOMENT MATRIX, M
81 -    Ix2 = conv2(double(ix.^2),G,'same');
82 -    Iy2 = conv2(double(iy.^2),G,'same');
83 -    Ixy = conv2(double(ix.*iy),G,'same');
84      % CORNERNESS MEASURE
85 -    r = (Ix2.*Iy2 - Ixy.^2)./(Ix2+Iy2 + eps);
86      % Value of Threshold set empirically
87 -    threshold = 4000;
88      % FINFDING MAX POINT FOR NON-MAX SUPPRESSSION
89 -    maximum_point = ordfilt2(r, order^2,ones(order));
90      % FINDING CORNERS
91 -    harris_corners = (r==maximum_point) & (r>threshold);
92 -    [R,C]=find(harris_corners);
93 -    end
94
```

Important Note:

Repeatability of matched keypoints is checked against two values of threshold. In first observation it is set to be 100 i.e. keeping it minimum as it should be ideally, as the distance between predicted keypoint and the one in rotated image should be close to each other. In second observation the threshold value is set the same as used in Harris corner detector in Part-A and for some reason it's giving a 100 percent repeatability. Although, it is not supported logically. The repeatability plots for both observations can be seen in the results section.

Observation # 01

Threshold value is set to be 100 and the Euclidean distance is compared with it as shown in the code snippet below. Plots showing corners can be seen in the results section. The number of matched keypoints comes out be less than the keypoints in original image

Code snippet

```
34
35          % PREDICTING IDEAL POSITION OF KEYPOINT
36 -        thresh = 100;
37 -    for i=1:1:Ocpointssize
38          % KEYPOINT ROTATED BY TRANSFORMATION MATRIX
39 -        RotatedP = RotMatrix*((Ocpoints(i))'-CenterO)+CenterR;
40          % EUCLIDEAN DISTANCE BTW TRANSFORMED & ROTATED IMAGE KEYPOINT
41 -        Euc_D = sqrt(sum((Rcpoints(i)-RotatedP).^2));
42 -        if(Euc_D<thresh) % IF DISTANCE LESS THAN THRESHOLD, THEN
43 -        M=M+1; % INCREMENT IN COUNT OF MATCHED KEYPOINTS
44 -        end
45 -    end
46
```

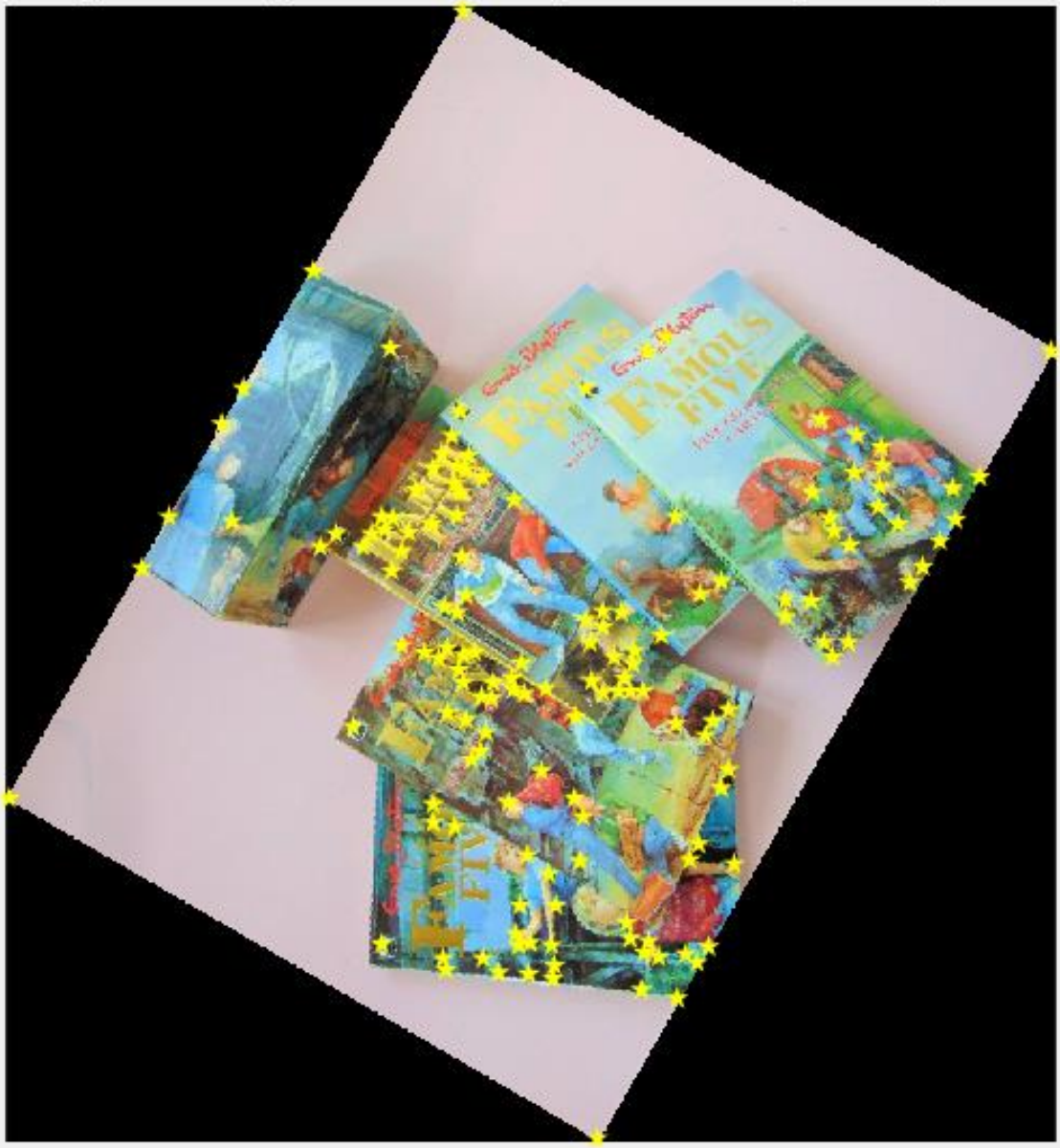
Results

Rotated image with 15 degree where M = 36 , N = 152 and Repeatability M/N = 0.23684

Rotated image with 30 degree where M = 42 , N = 152 and Repeatability M/N = 0.27632

Rotated image with 45 degree where M = 31 , N = 152 and Repeatability M/N = 0.20395

Rotated image with 60 degree where M = 21 , N = 152 and Repeatability M/N = 0.13816

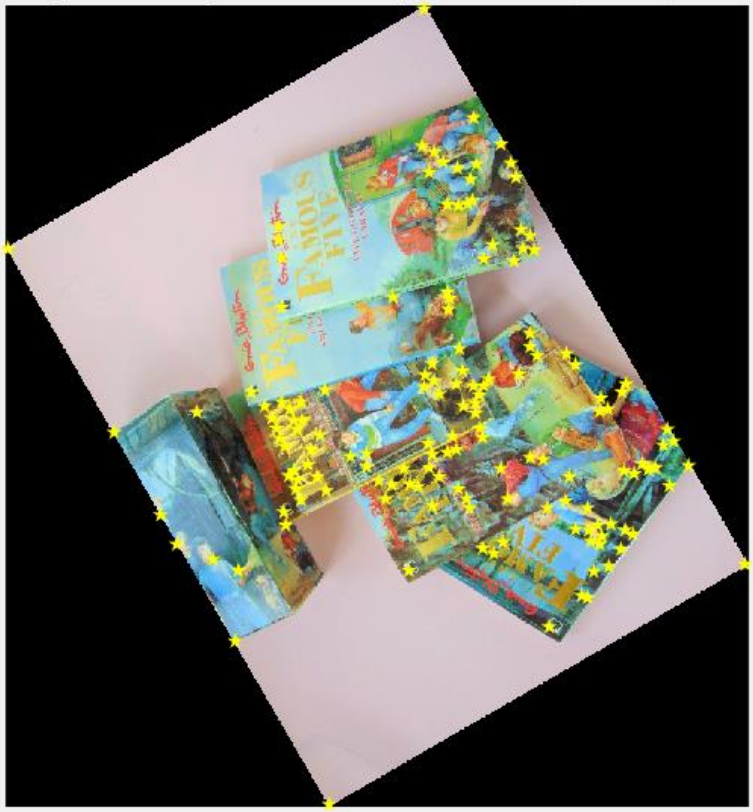


Rotated image with 75 degree where M = 13 , N = 152 and Repeatability M/N = 0.085526

Rotated image with 90 degree where M = 0 , N = 152 and Repeatability M/N = 0

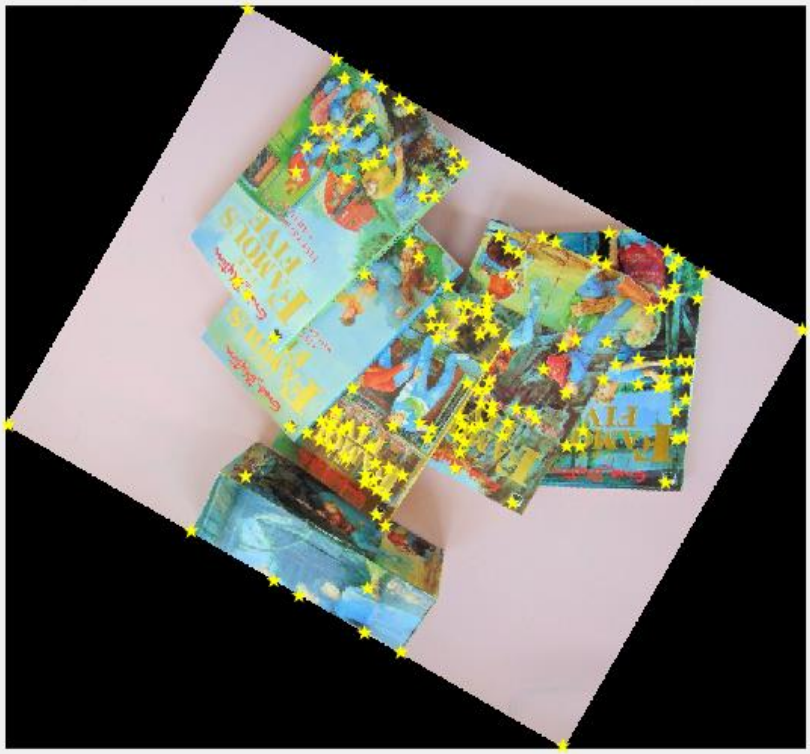
Rotated image with 105 degree where M = 9 , N = 152 and Repeatability M/N = 0.059211

Rotated image with 120 degree where M = 31 , N = 152 and Repeatability M/N = 0.20395



Rotated image with 135 degree where M = 40 , N = 152 and Repeatability M/N = 0.26316

Rotated image with 150 degree where M = 35 , N = 152 and Repeatability M/N = 0.23026

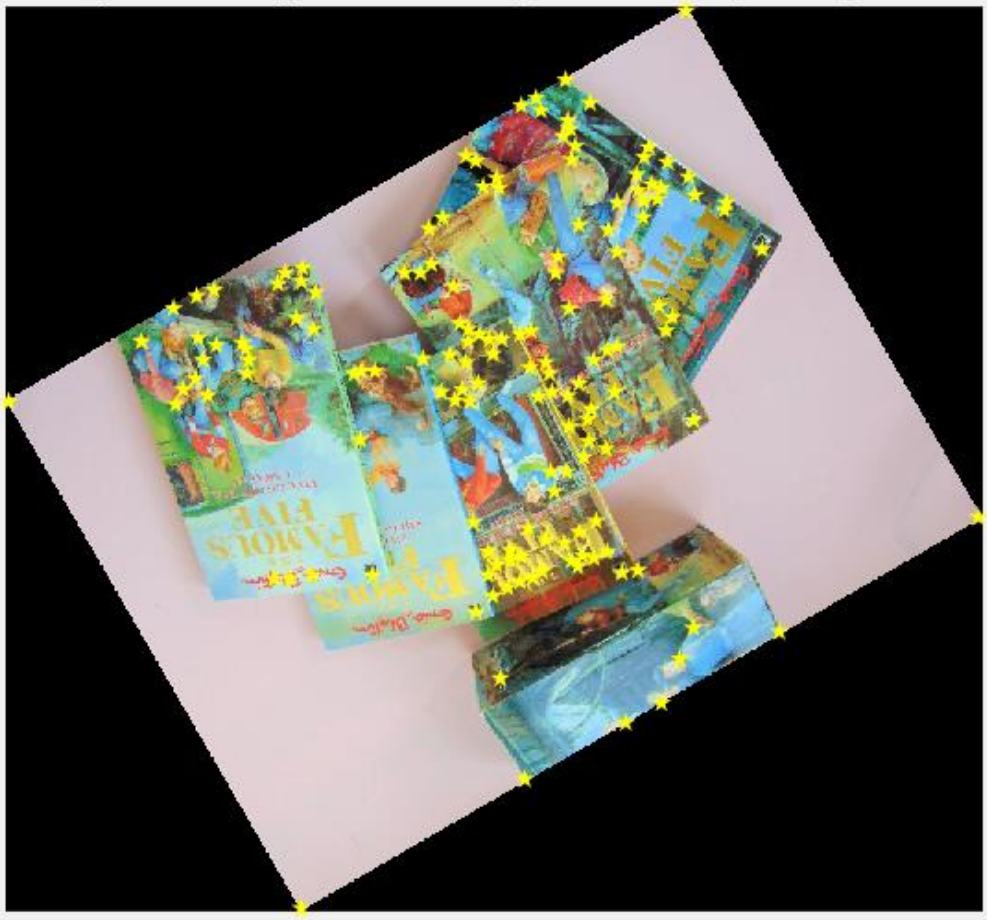


Rotated image with 165 degree where M = 41 , N = 152 and Repeatability M/N = 0.26974

Rotated image with 180 degree where M = 46 , N = 152 and Repeatability M/N = 0.30263

Rotated image with 195 degree where M = 28 , N = 152 and Repeatability M/N = 0.18421

Rotated image with 210 degree where M = 36 , N = 152 and Repeatability M/N = 0.23684



Rotated image with 225 degree where M = 31 , N = 152 and Repeatability M/N = 0.20395

Rotated image with 240 degree where M = 34 , N = 152 and Repeatability M/N = 0.22368

Rotated image with 255 degree where M = 6 , N = 152 and Repeatability M/N = 0.039474

Rotated image with 270 degree where M = 0 , N = 152 and Repeatability M/N = 0

Rotated image with 285 degree where M = 5 , N = 152 and Repeatability M/N = 0.032895

Rotated image with 300 degree where M = 28 , N = 152 and Repeatability M/N = 0.18421

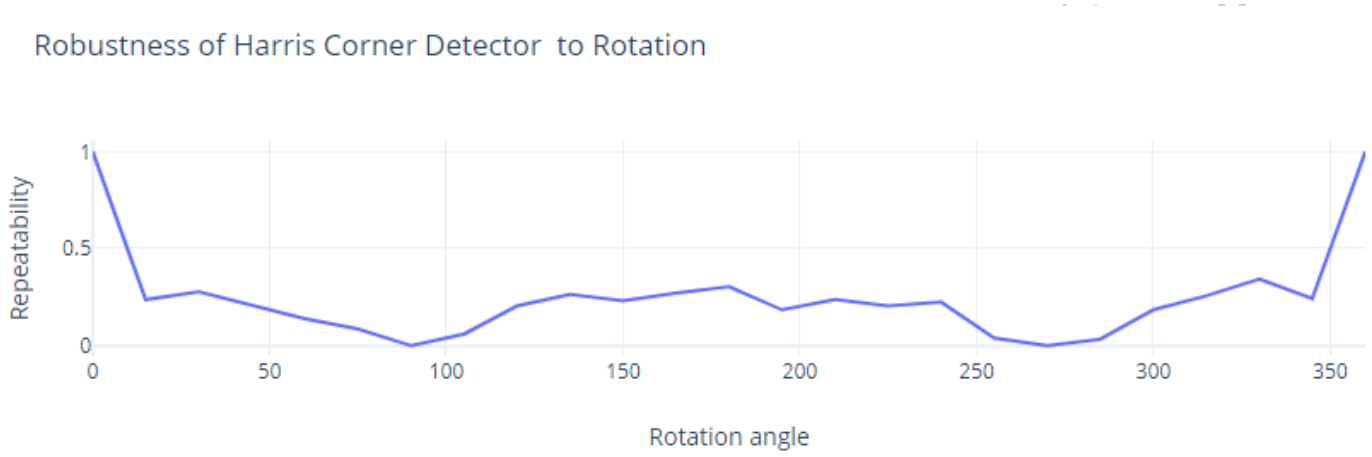
Rotated image with 315 degree where M = 39 , N = 152 and Repeatability M/N = 0.25658

Rotated image with 330 degree where M = 52 , N = 152 and Repeatability M/N = 0.34211

Rotated image with 345 degree where M = 37 , N = 152 and Repeatability M/N = 0.24342

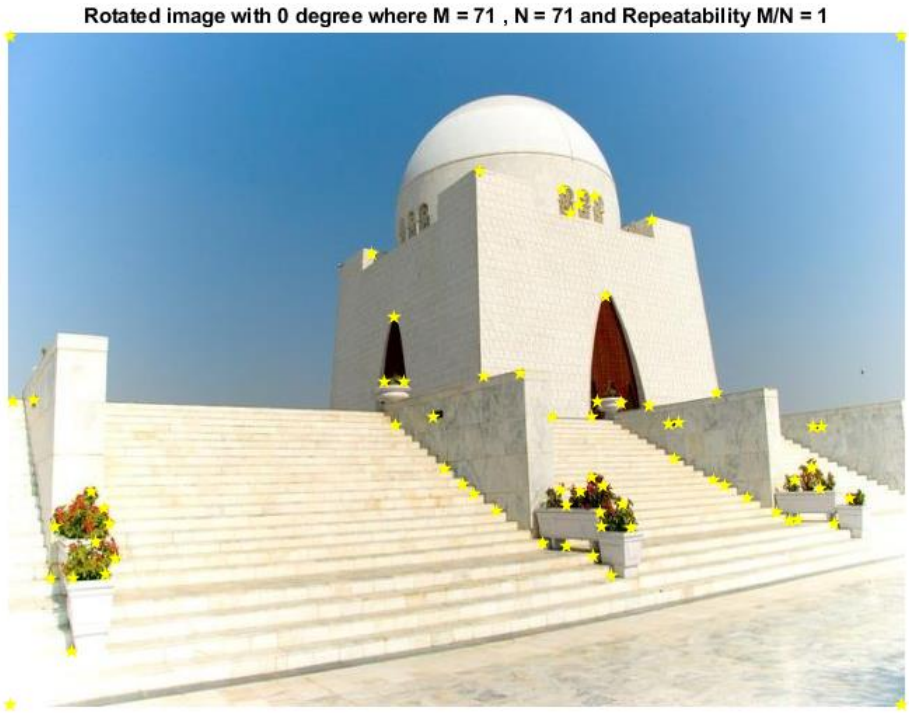
Rotated image with 360 degree where M = 152 , N = 152 and Repeatability M/N = 1

Repeatability vs Rotation Plot for the Image Famous Five



Conclusion

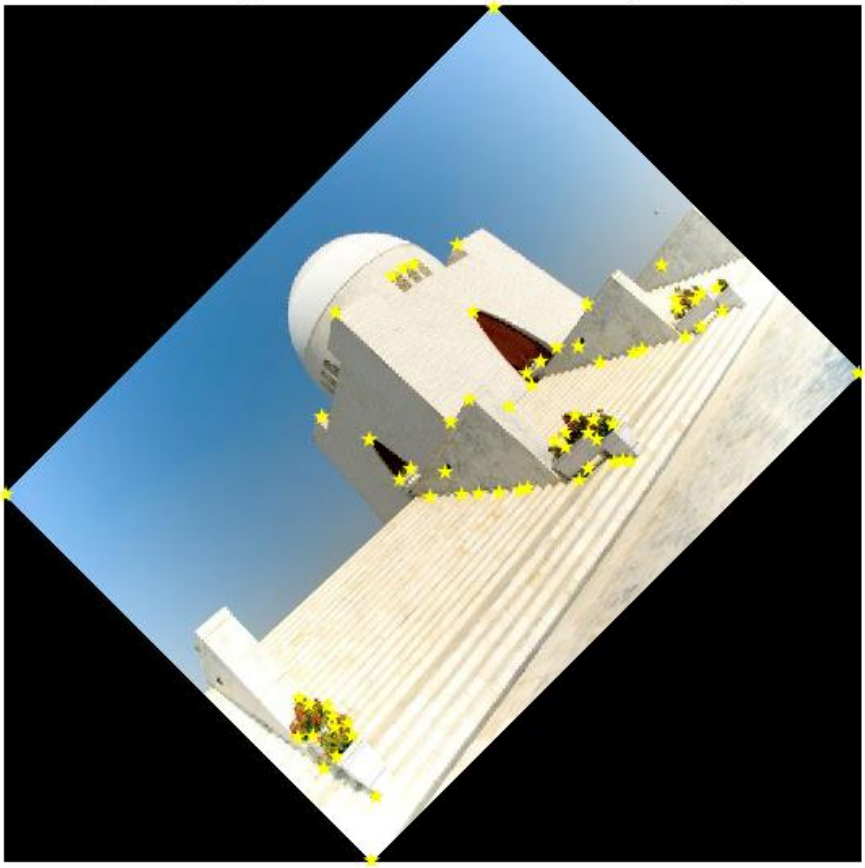
Ideally, the repeatability plot for rotation must maintain the ratio at 0.95 but due to some unknown reason or inefficient programming, it falls as soon it gets rotated and couldn't locate the rotated keypoints. Harris detector is robust to rotation but here the results are not so favorable.



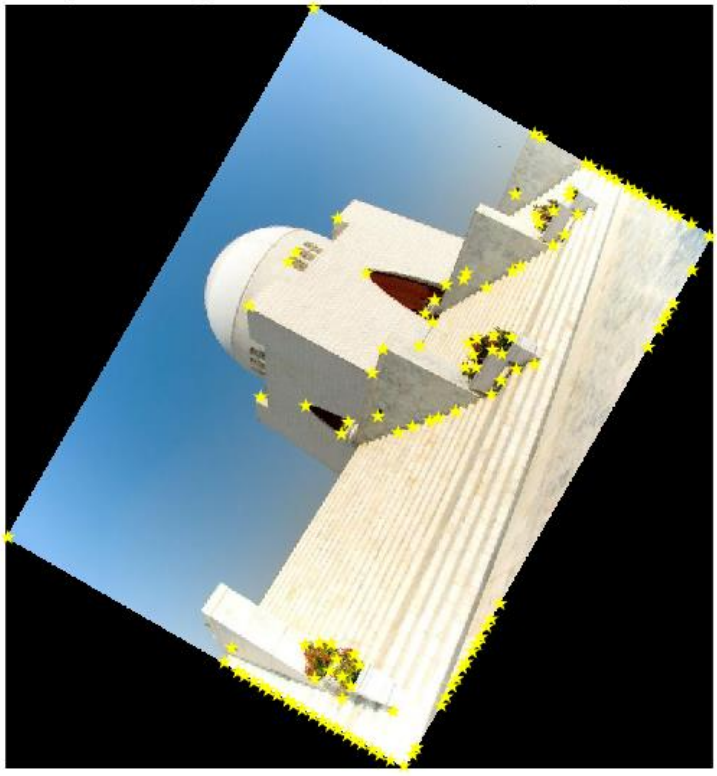
Rotated image with 15 degree where M = 18 , N = 71 and Repeatability M/N = 0.25352

Rotated image with 30 degree where M = 14 , N = 71 and Repeatability M/N = 0.19718

Rotated image with 45 degree where M = 16 , N = 71 and Repeatability M/N = 0.22535



Rotated image with 60 degree where M = 9 , N = 71 and Repeatability M/N = 0.12676

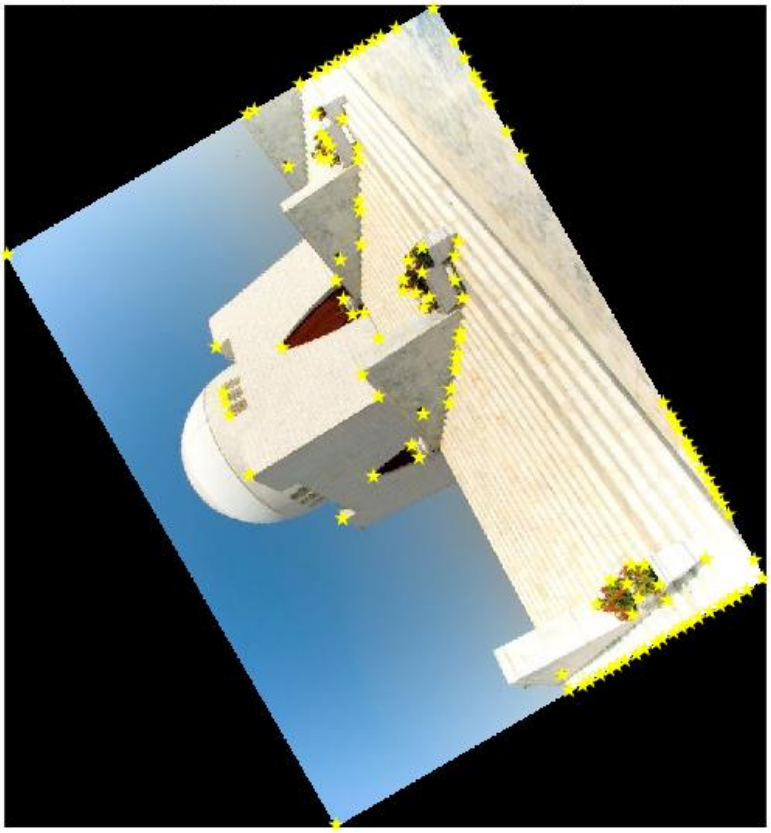


Rotated image with 75 degree where M = 5 , N = 71 and Repeatability M/N = 0.070423

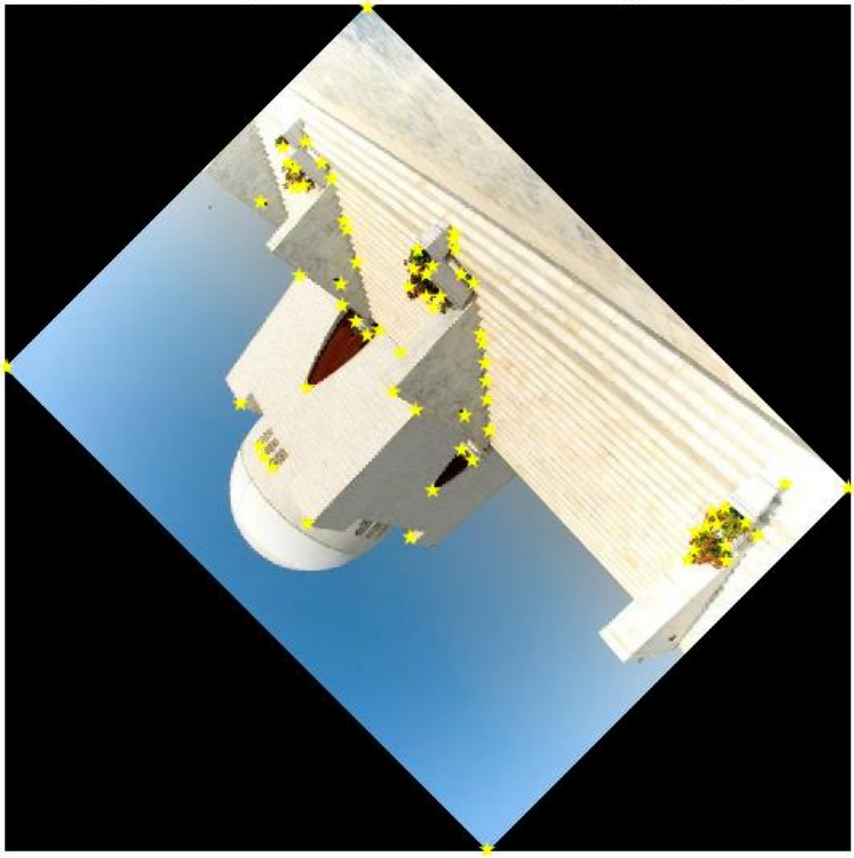
Rotated image with 90 degree where M = 0 , N = 71 and Repeatability M/N = 0

Rotated image with 105 degree where M = 2 , N = 71 and Repeatability M/N = 0.028169

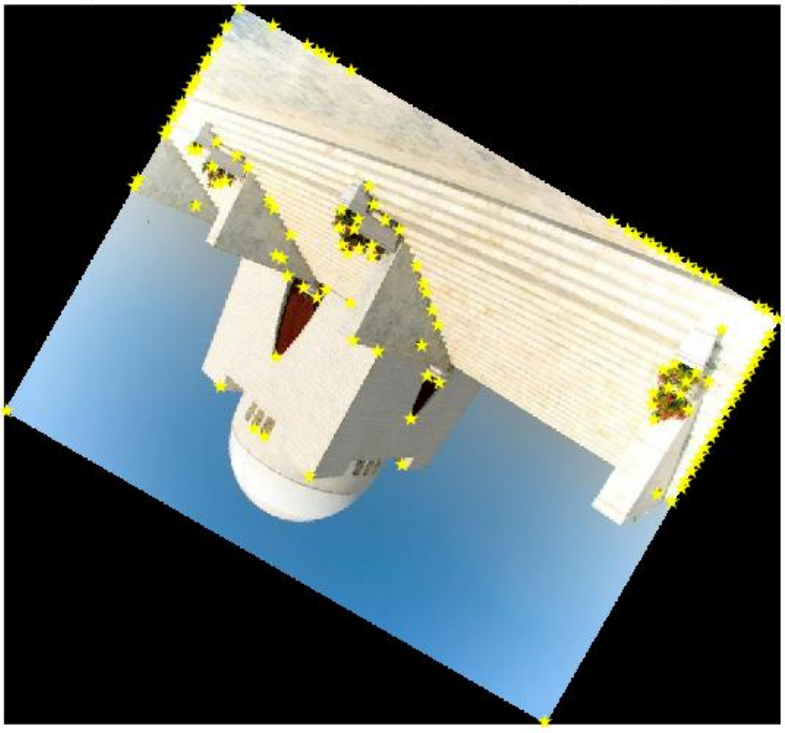
Rotated image with 120 degree where M = 9 , N = 71 and Repeatability M/N = 0.12676



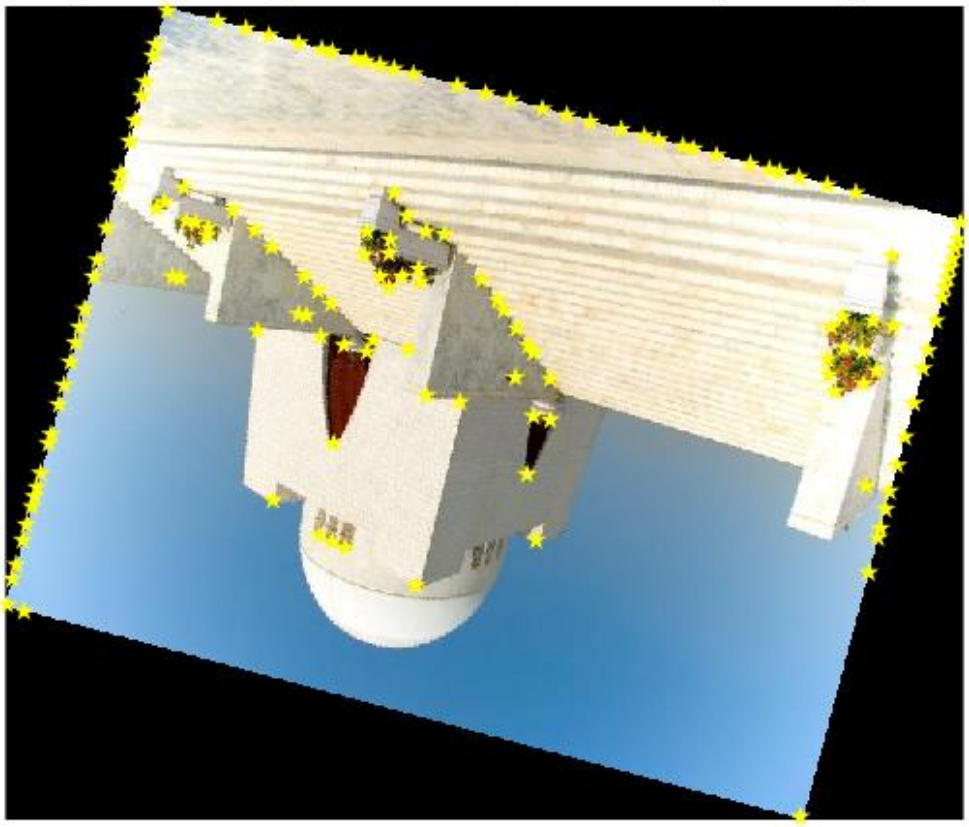
Rotated image with 135 degree where M = 22 , N = 71 and Repeatability M/N = 0.30986



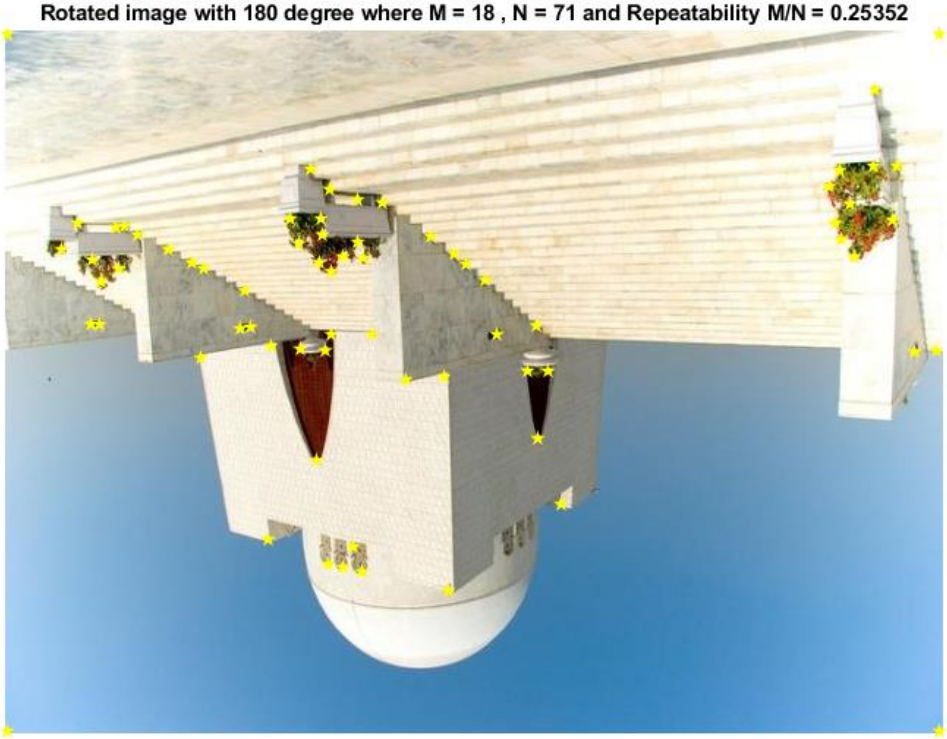
Rotated image with 150 degree where M = 11 , N = 71 and Repeatability M/N = 0.15493



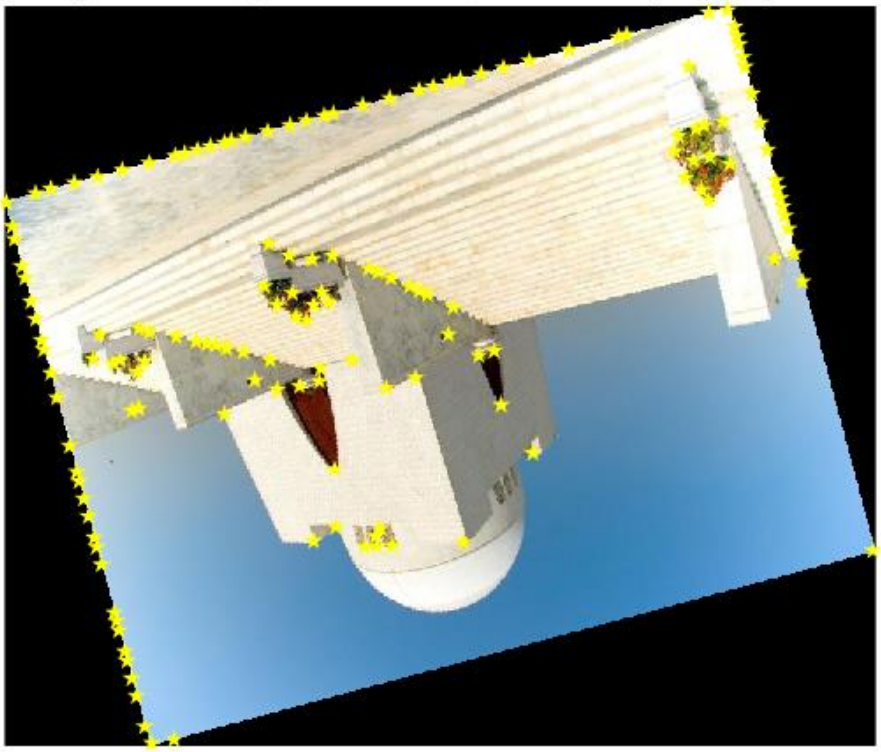
Rotated image with 165 degree where M = 21 , N = 71 and Repeatability M/N = 0.29577



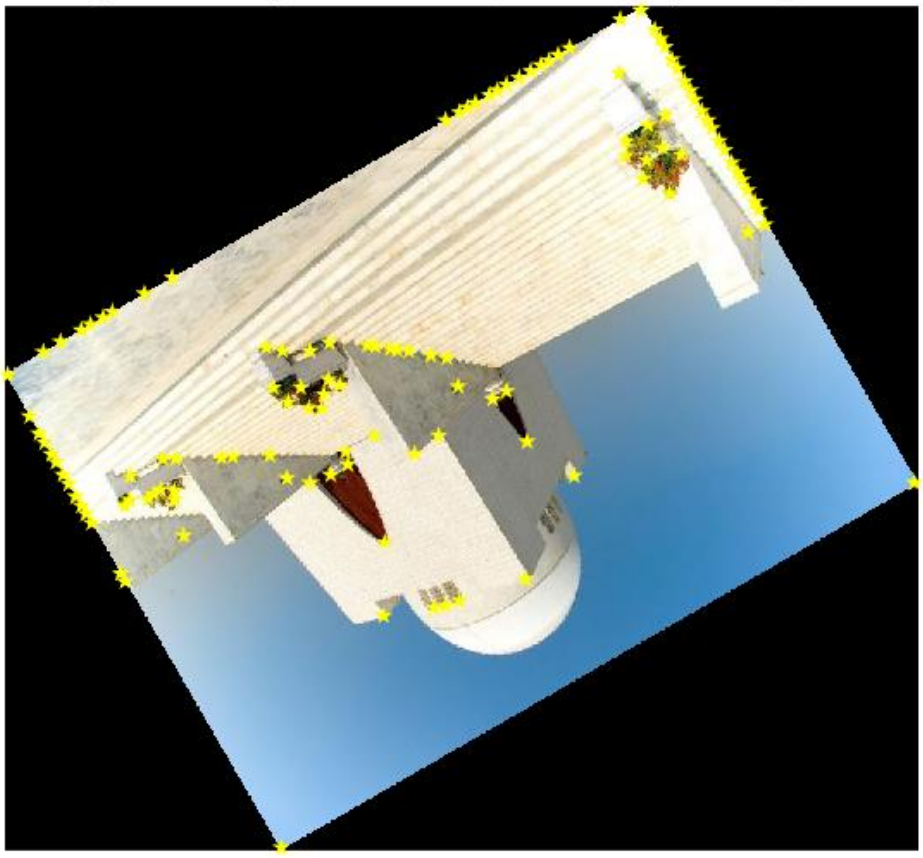
Rotated image with 180 degree where M = 18 , N = 71 and Repeatability M/N = 0.25352



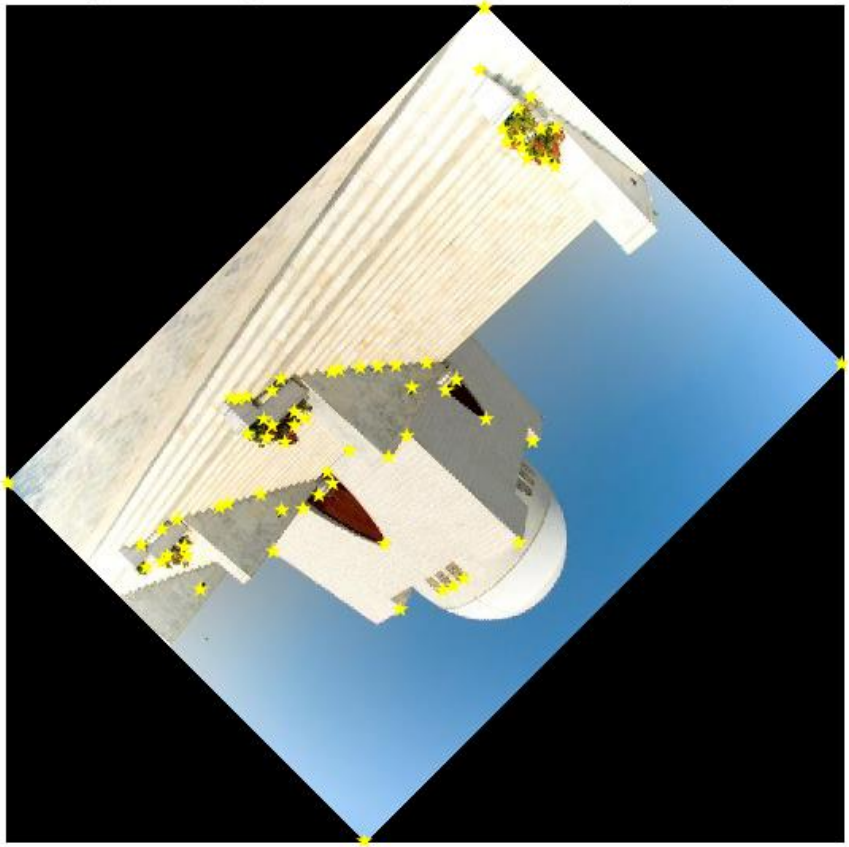
Rotated image with 195 degree where M = 9 , N = 71 and Repeatability M/N = 0.12676



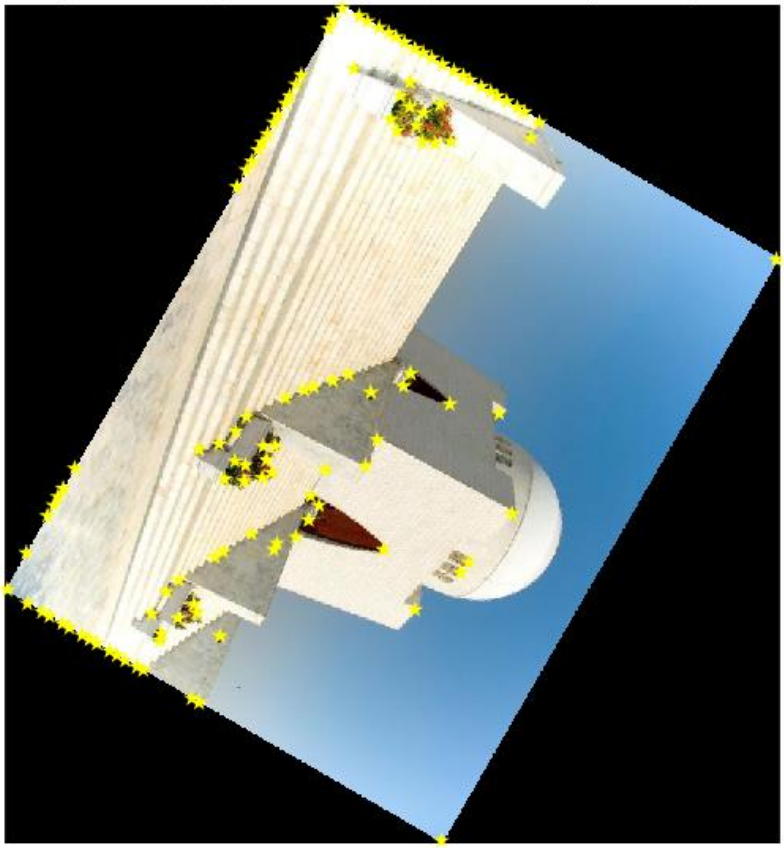
Rotated image with 210 degree where M = 30 , N = 71 and Repeatability M/N = 0.42254



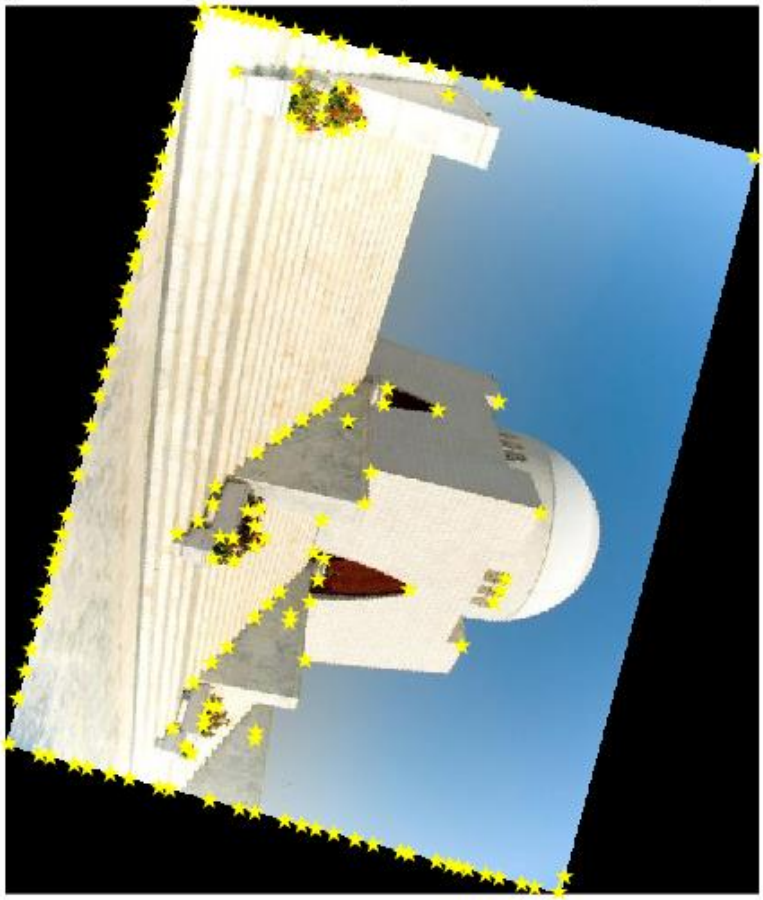
Rotated image with 225 degree where M = 21 , N = 71 and Repeatability M/N = 0.29577



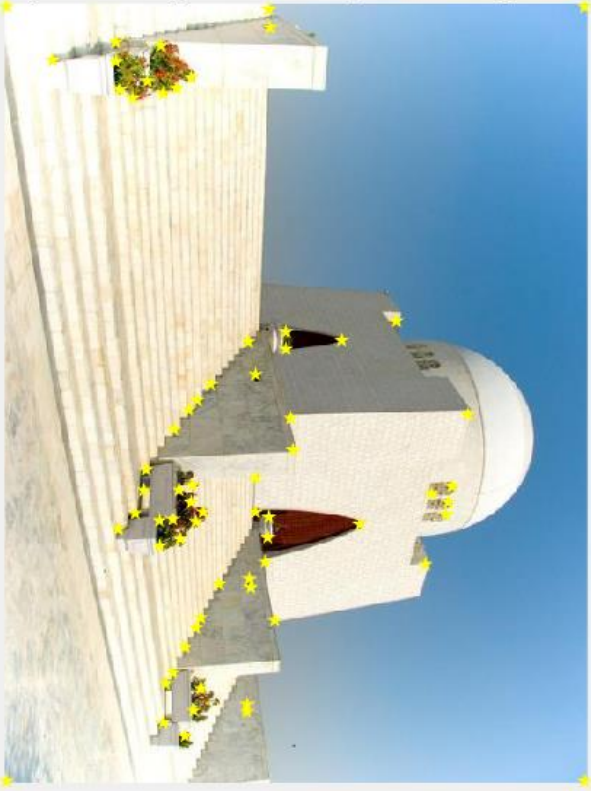
Rotated image with 240 degree where M = 14 , N = 71 and Repeatability M/N = 0.19718



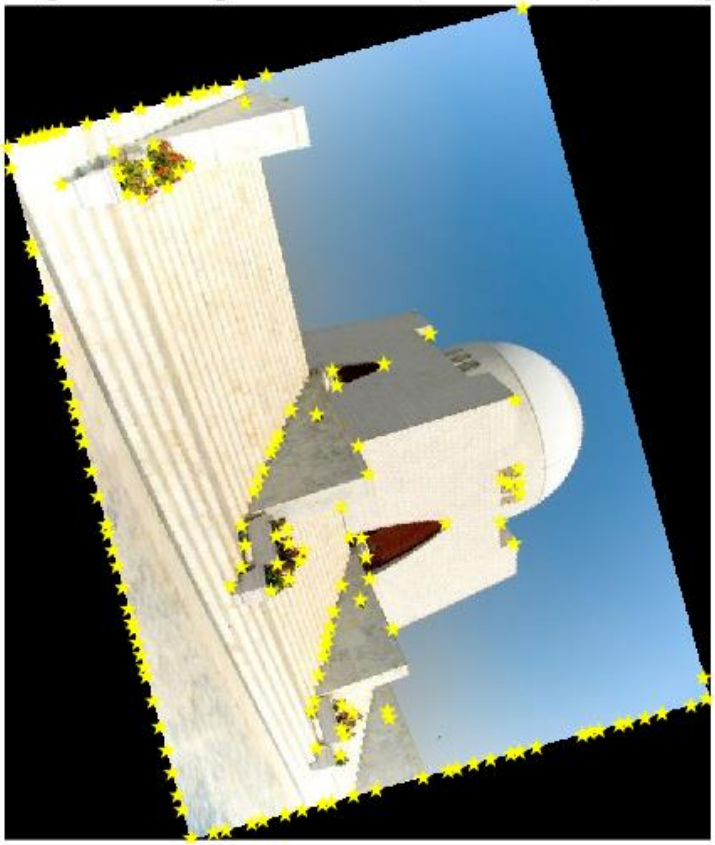
Rotated image with 255 degree where M = 6 , N = 71 and Repeatability M/N = 0.084507



Rotated image with 270 degree where M = 0 , N = 71 and Repeatability M/N = 0



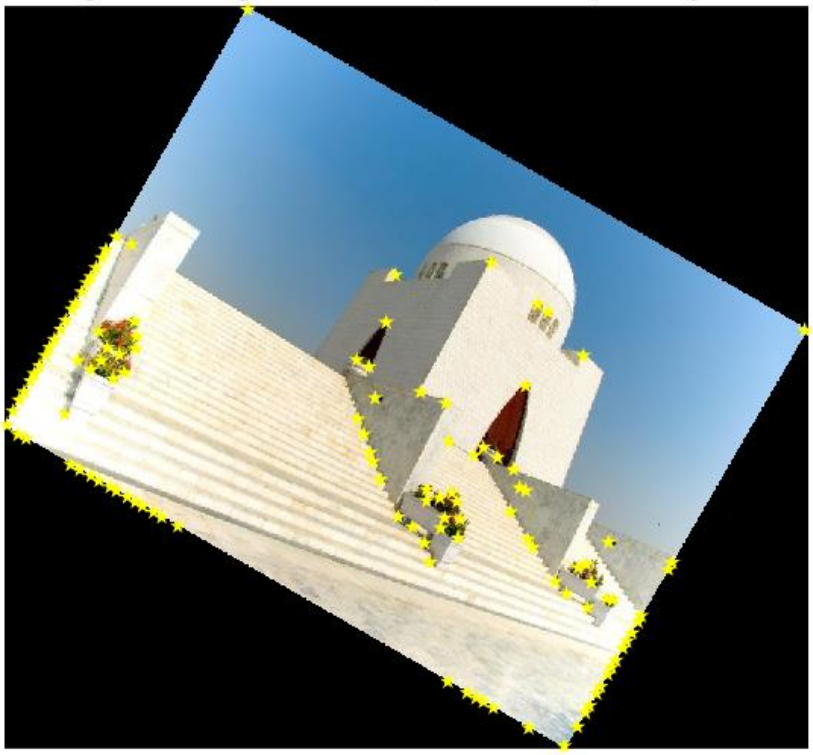
Rotated image with 285 degree where M = 0 , N = 71 and Repeatability M/N = 0



Rotated image with 300 degree where M = 16 , N = 71 and Repeatability M/N = 0.22535

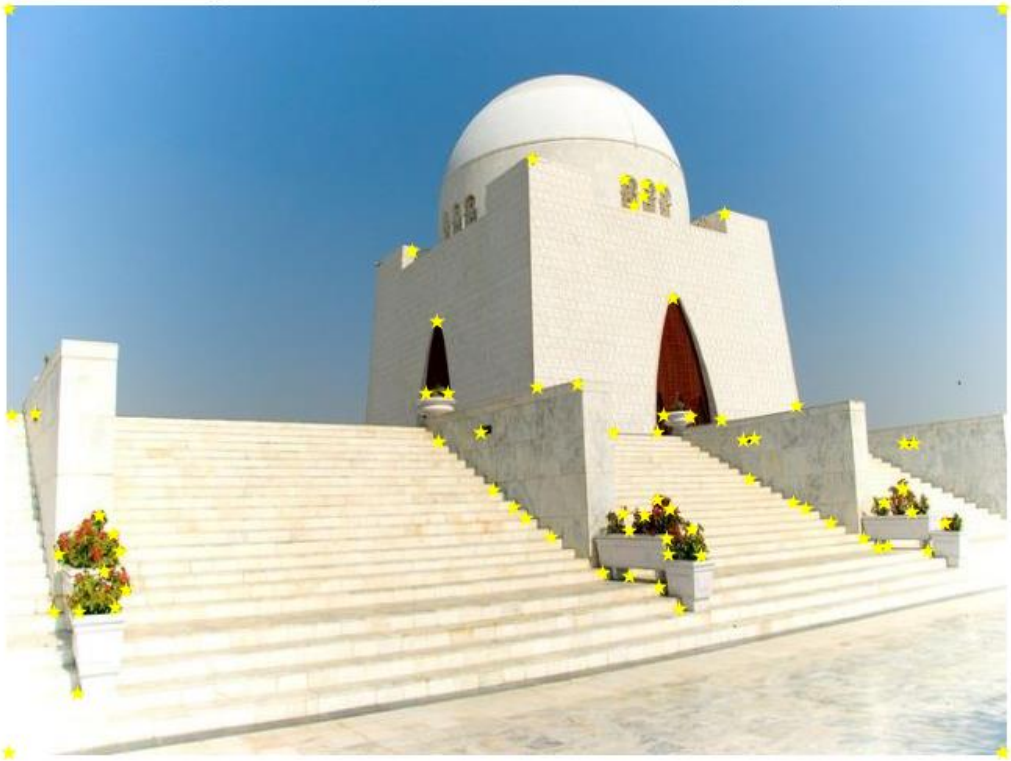
Rotated image with 315 degree where M = 9 , N = 71 and Repeatability M/N = 0.12676

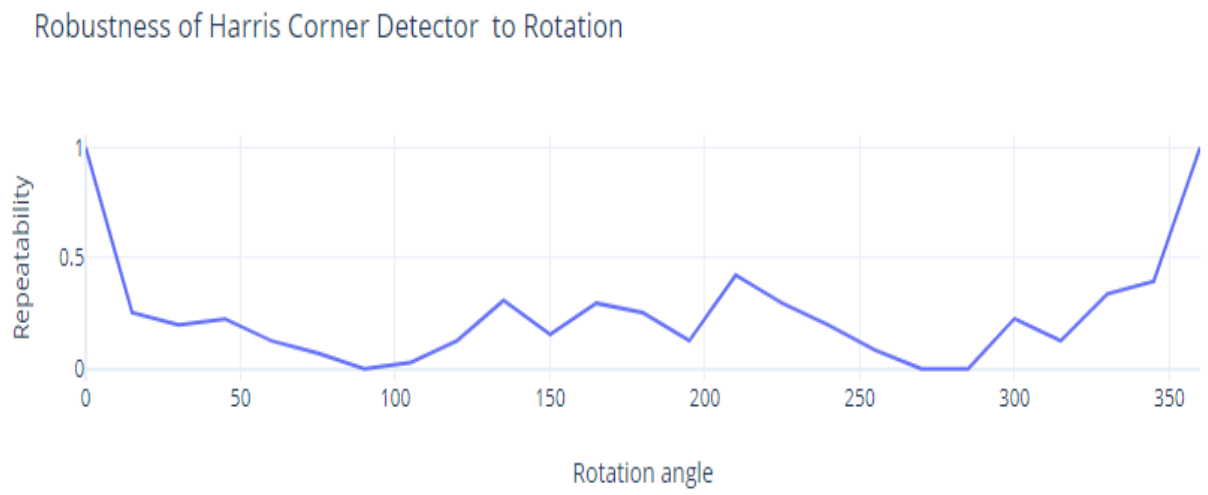
Rotated image with 330 degree where M = 24 , N = 71 and Repeatability M/N = 0.33803



Rotated image with 345 degree where M = 28 , N = 71 and Repeatability M/N = 0.39437

Rotated image with 360 degree where M = 71 , N = 71 and Repeatability M/N = 1





Conclusion

For the second image we get almost the same result with slight difference in values but the overall pattern is same. Ideally, the repeatability plot for rotation must maintain the ratio at 0.95 but due to some unknown reason or inefficient programming, it falls as soon it gets rotated and couldn't locate the rotated keypoints. Harris detector is robust to rotation but here the results are not so favorable.

Observation # 02

The threshold value is set to be equal to that used in Harris corner detector in Part- A i.e. 4000. It was observed that number of matched corners (M) comes out to be equal to corners in original image (N), so, we get a repeatability factor equal to 1. It can be seen in the plot of repeatability vs rotation in the results section.

P.s. Ideally threshold value must have low value e.g. 60 or 80 when finding the keypoint in the rotated image that matches the ones in original image. Displaying the results for T =4000 only as a finding whose reasoning couldn't be found out.

Code snippet

```
34
35          % PREDICTING IDEAL POSITION OF KEYPOINT
36 -        threshold = 4000;
37 -    for i=1:1:Ocpointssize
38          % KEYPOINT ROTATED BY TRANSFORMATION MATRIX
39 -        RotatedP = RotMatrix*((Ocpoints(i))'-CenterO)+CenterR;
40          % EUCLIDEAN DISTANCE BTW TRANSFORMED & ROTATED IMAGE KEYPOINT
41 -        Euc_D = sqrt(sum((Rcpoints(i)-RotatedP).^2));
42 -        if(Euc_D<threshold) % IF DISTANCE LESS THAN THRESHOLD, THEN
43 -        M=M+1; % INCREMENT IN COUNT OF MATCHED KEYPOINTS
44 -        end
45 -    end
46
```

Rotated image with 15 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 30 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 45 degree where M = 152 , N = 152 and Repeatability M/N = 1

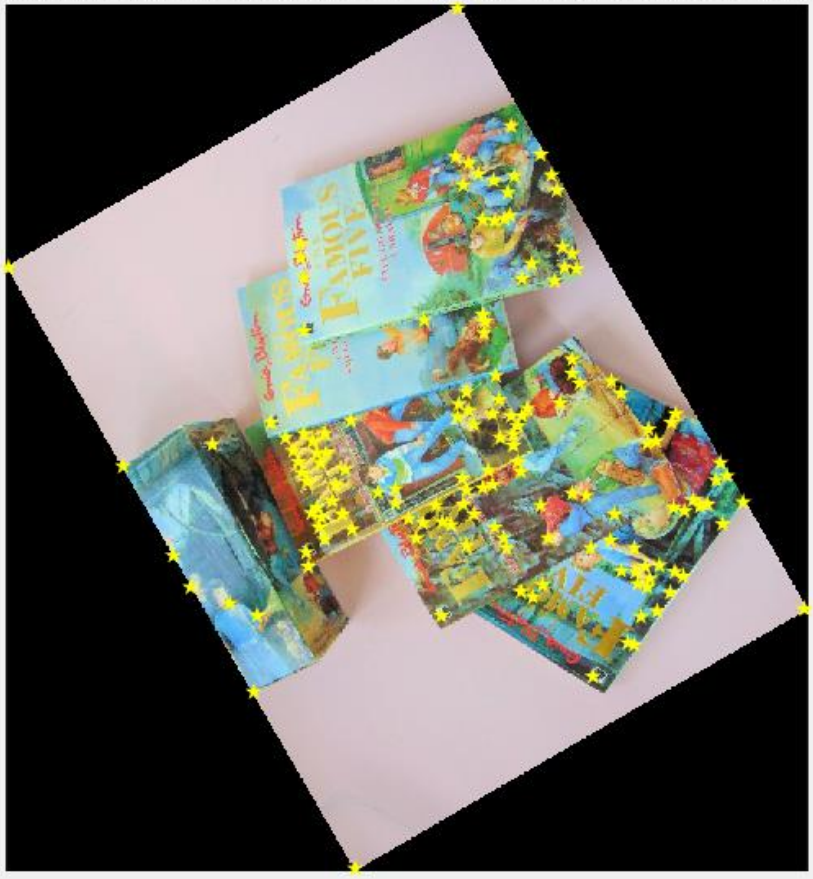
Rotated image with 60 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 75 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 90 degree where M = 152 , N = 152 and Repeatability M/N = 1

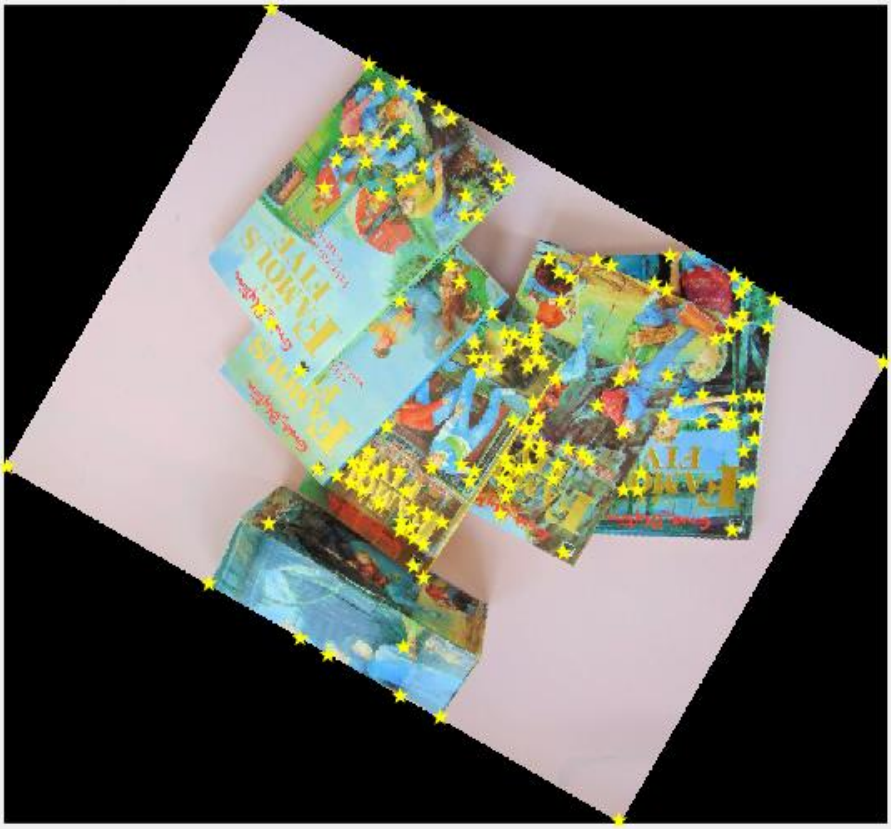
Rotated image with 105 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 120 degree where M = 152 , N = 152 and Repeatability M/N = 1



Rotated image with 135 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 150 degree where M = 152 , N = 152 and Repeatability M/N = 1

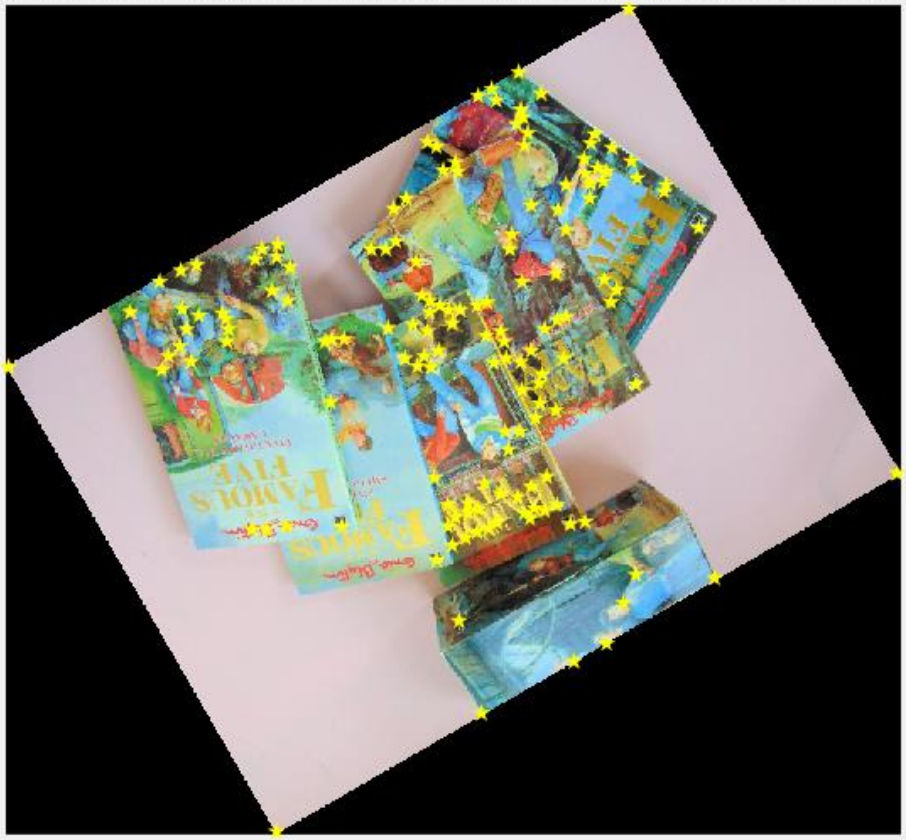


Rotated image with 165 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 180 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 195 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 210 degree where M = 152 , N = 152 and Repeatability M/N = 1



Rotated image with 225 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 240 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 255 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 270 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 285 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 300 degree where M = 152 , N = 152 and Repeatability M/N = 1

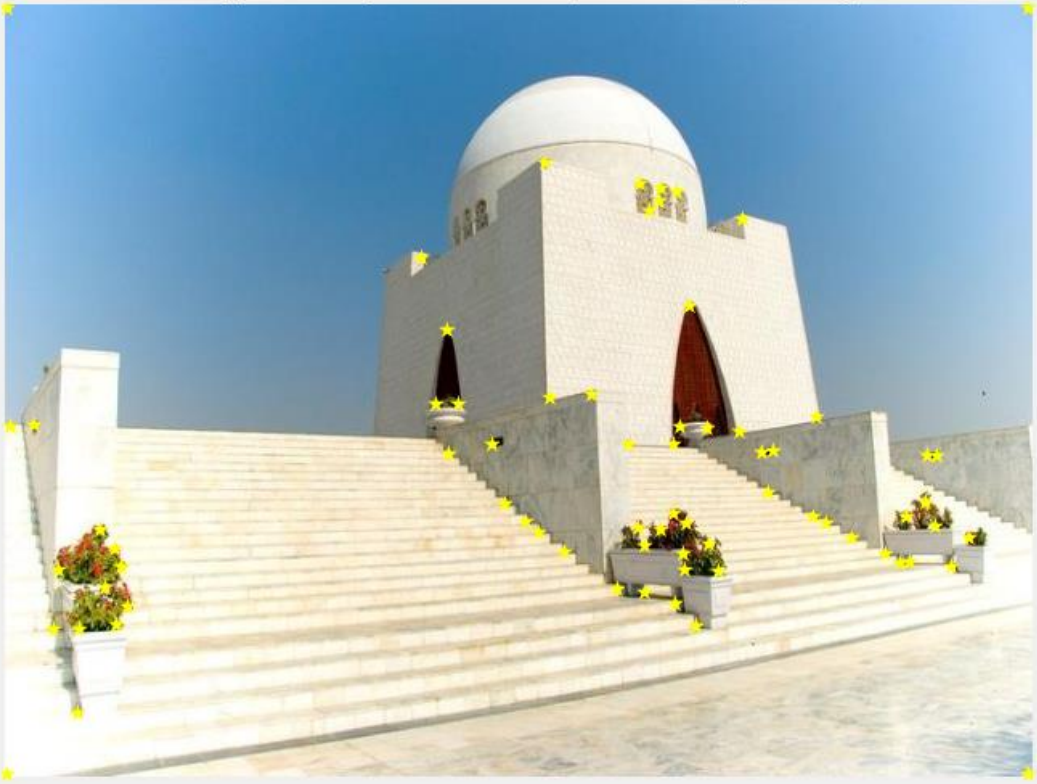
Rotated image with 315 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 330 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 345 degree where M = 152 , N = 152 and Repeatability M/N = 1

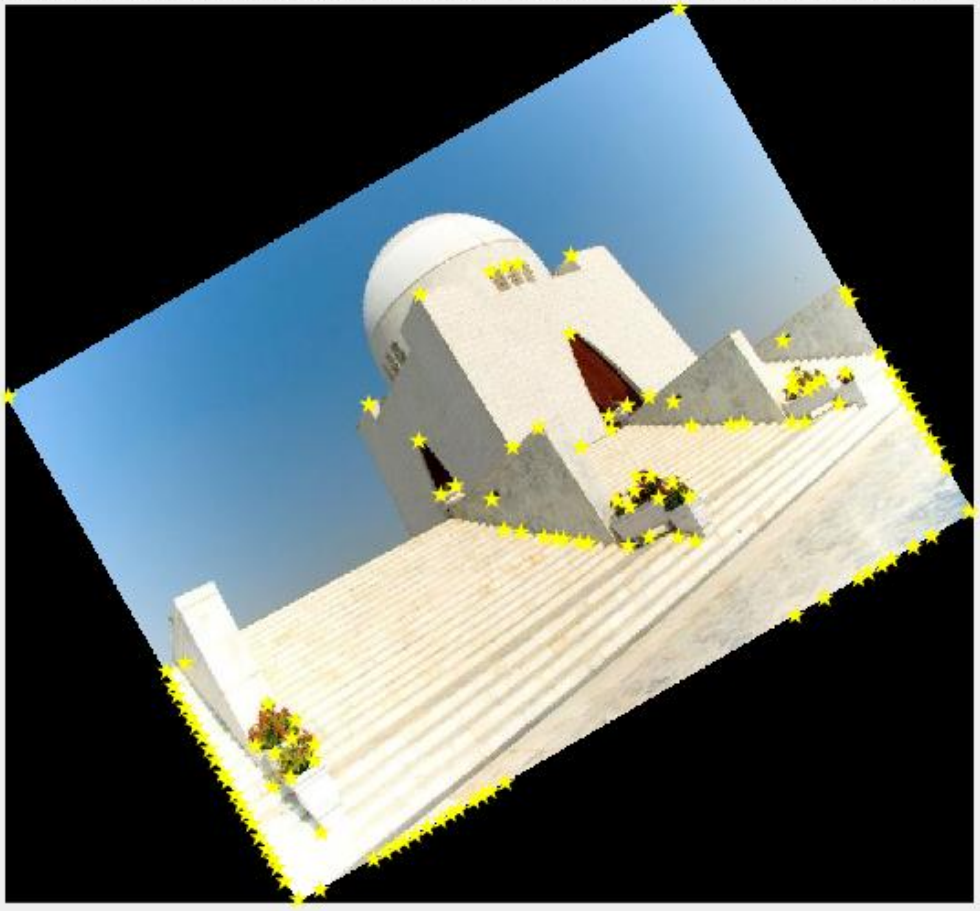
Rotated image with 360 degree where M = 152 , N = 152 and Repeatability M/N = 1

Rotated image with 0 degree where M = 71 , N = 71 and Repeatability M/N = 1

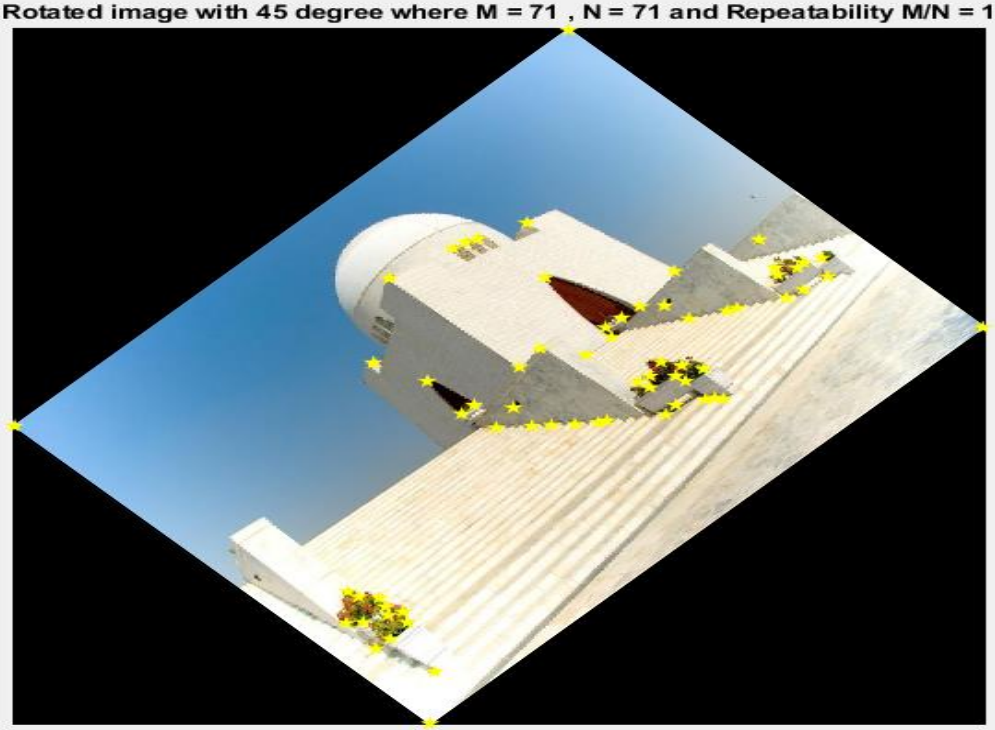


Rotated image with 15 degree where M = 71 , N = 71 and Repeatability M/N = 1

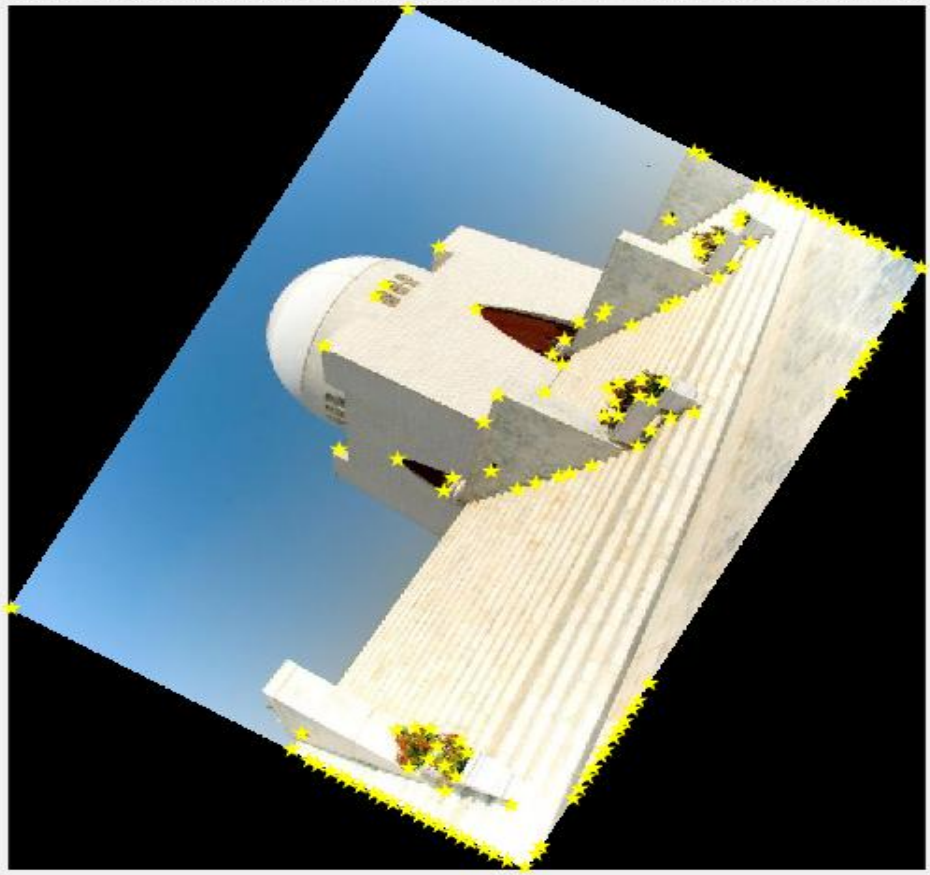
Rotated image with 30 degree where M = 71 , N = 71 and Repeatability M/N = 1



Rotated image with 45 degree where M = 71 , N = 71 and Repeatability M/N = 1



Rotated image with 60 degree where M = 71 , N = 71 and Repeatability M/N = 1

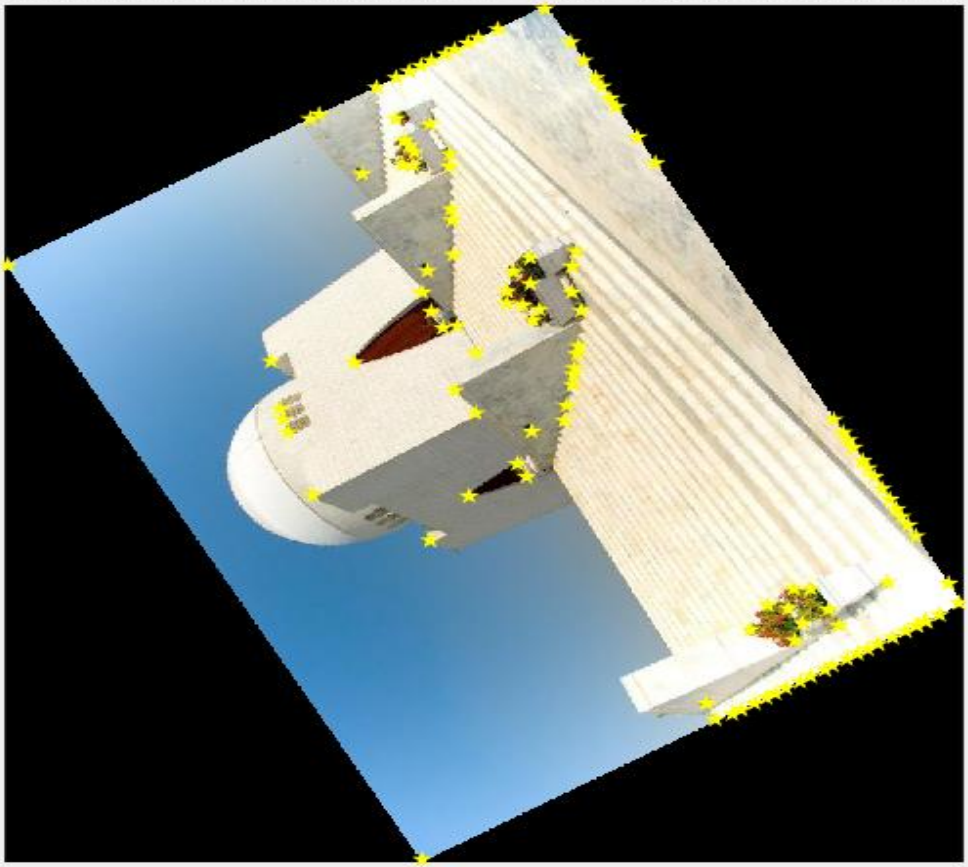


Rotated image with 75 degree where M = 71 , N = 71 and Repeatability M/N = 1

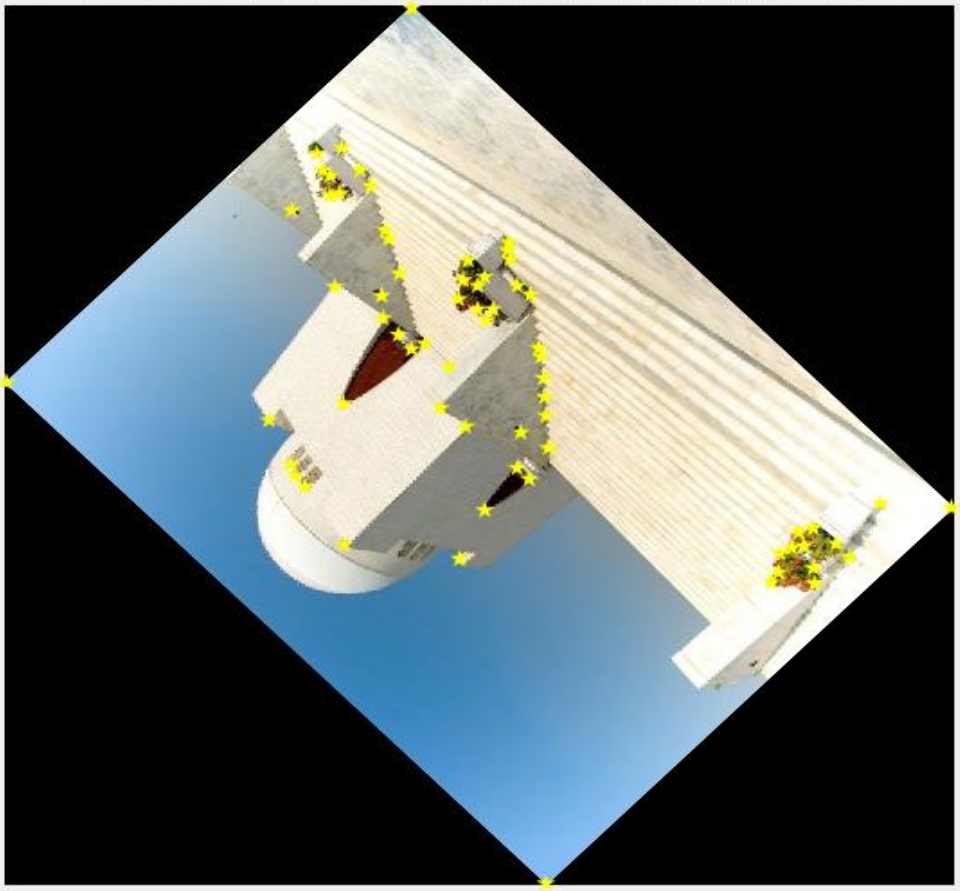
Rotated image with 90 degree where M = 71 , N = 71 and Repeatability M/N = 1

Rotated image with 105 degree where M = 71 , N = 71 and Repeatability M/N = 1

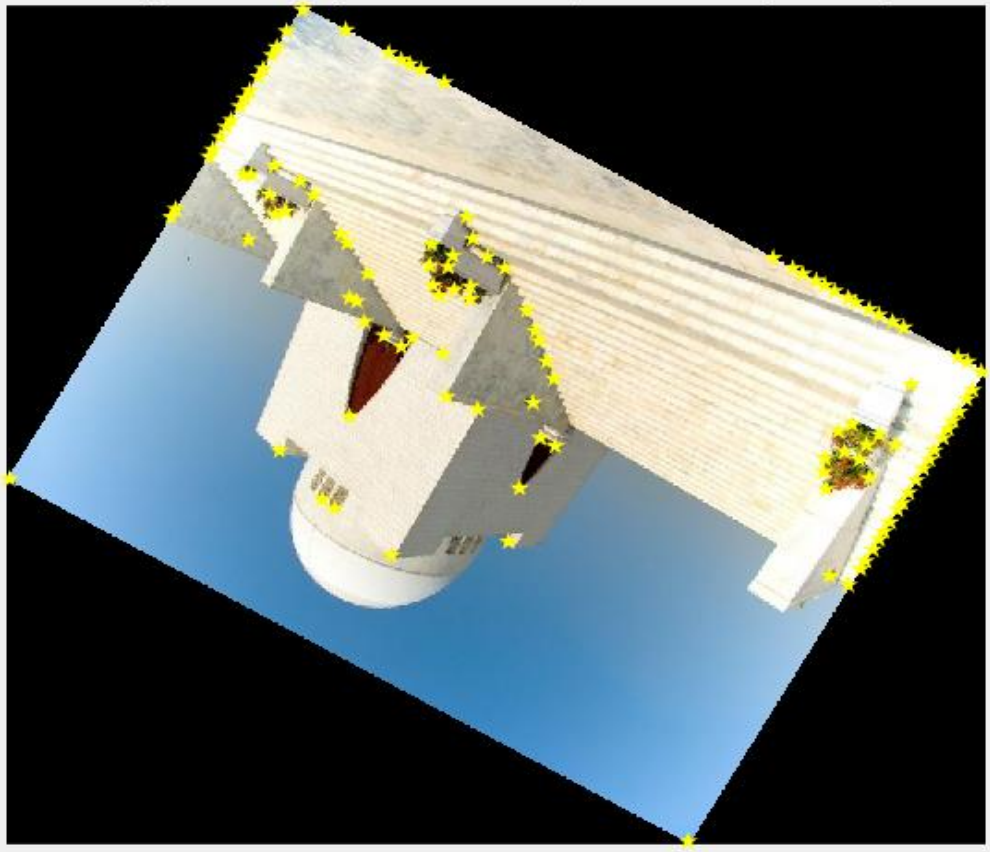
Rotated image with 120 degree where M = 71 , N = 71 and Repeatability M/N = 1



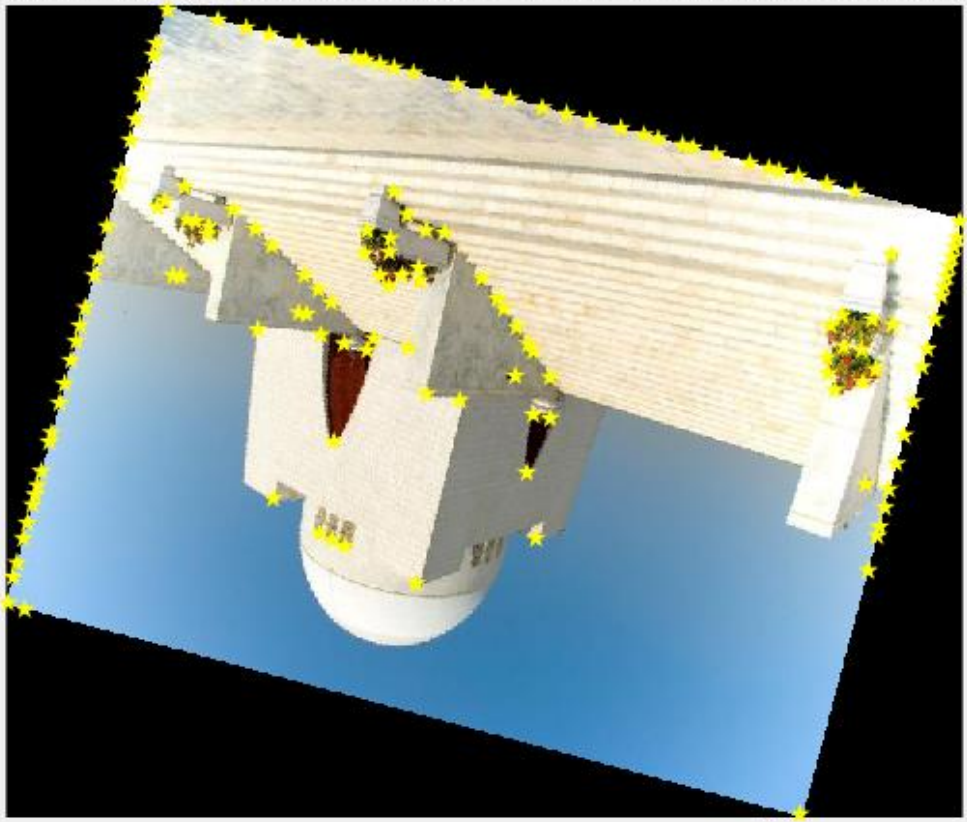
Rotated image with 135 degree where M = 71 , N = 71 and Repeatability M/N = 1



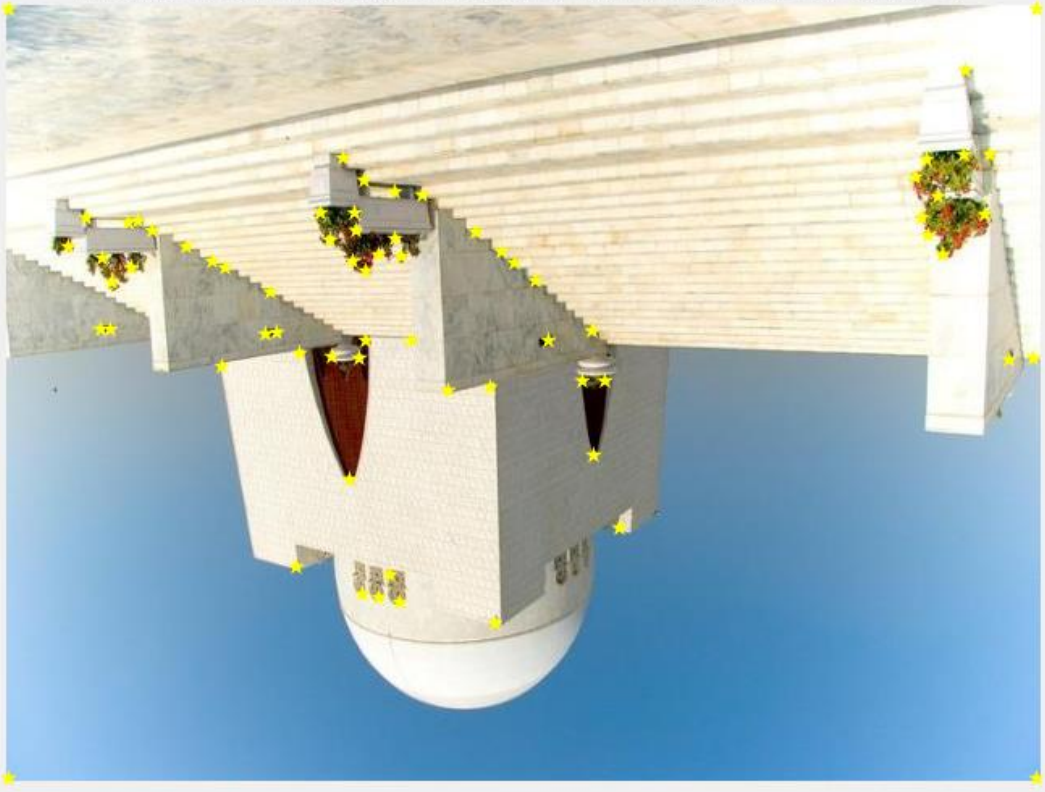
Rotated image with 150 degree where M = 71 , N = 71 and Repeatability M/N = 1



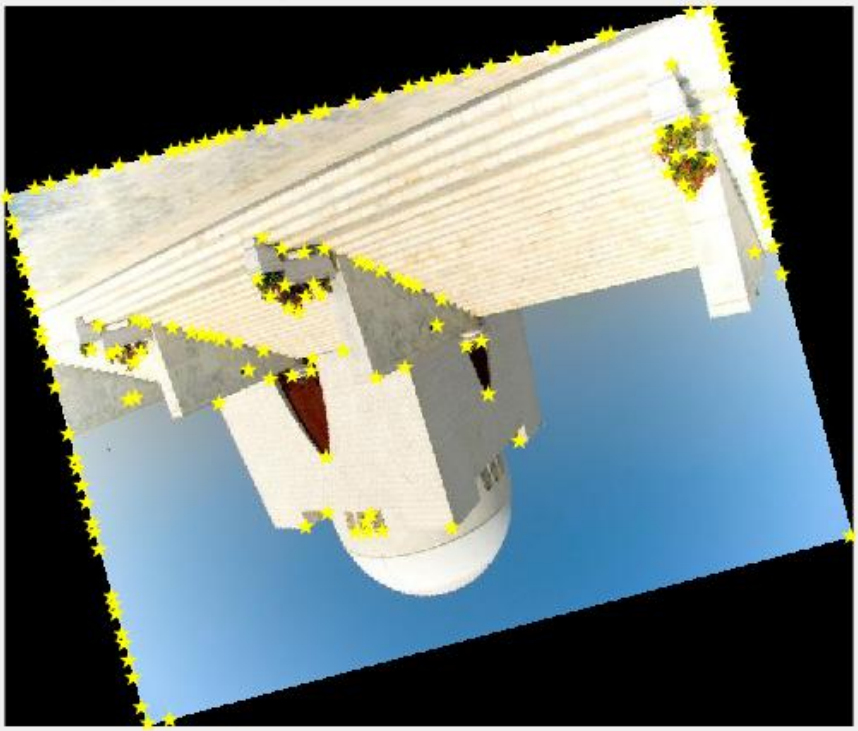
Rotated image with 165 degree where M = 71 , N = 71 and Repeatability M/N = 1



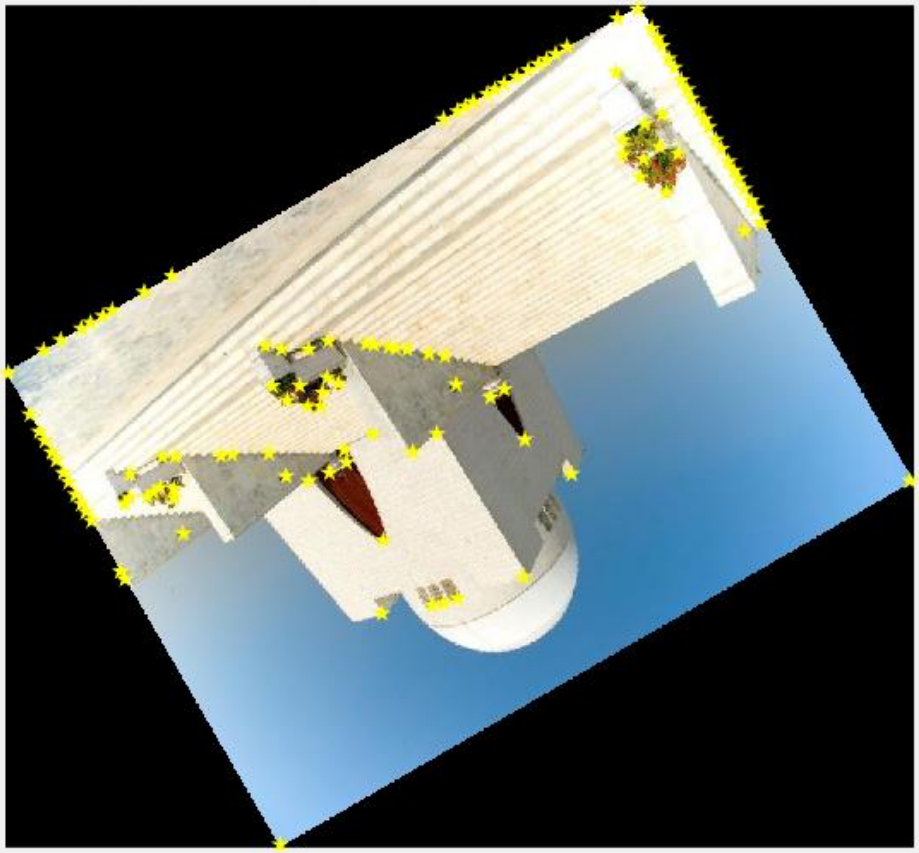
Rotated image with 180 degree where M = 71 , N = 71 and Repeatability M/N = 1



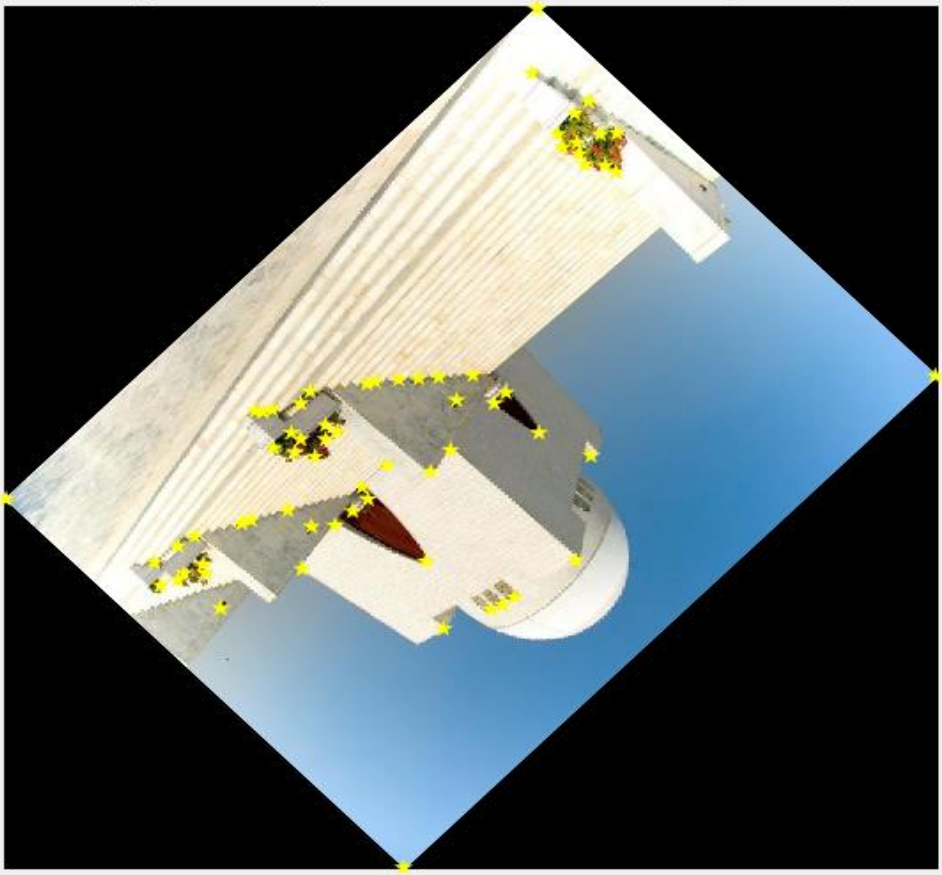
Rotated image with 195 degree where M = 71 , N = 71 and Repeatability M/N = 1



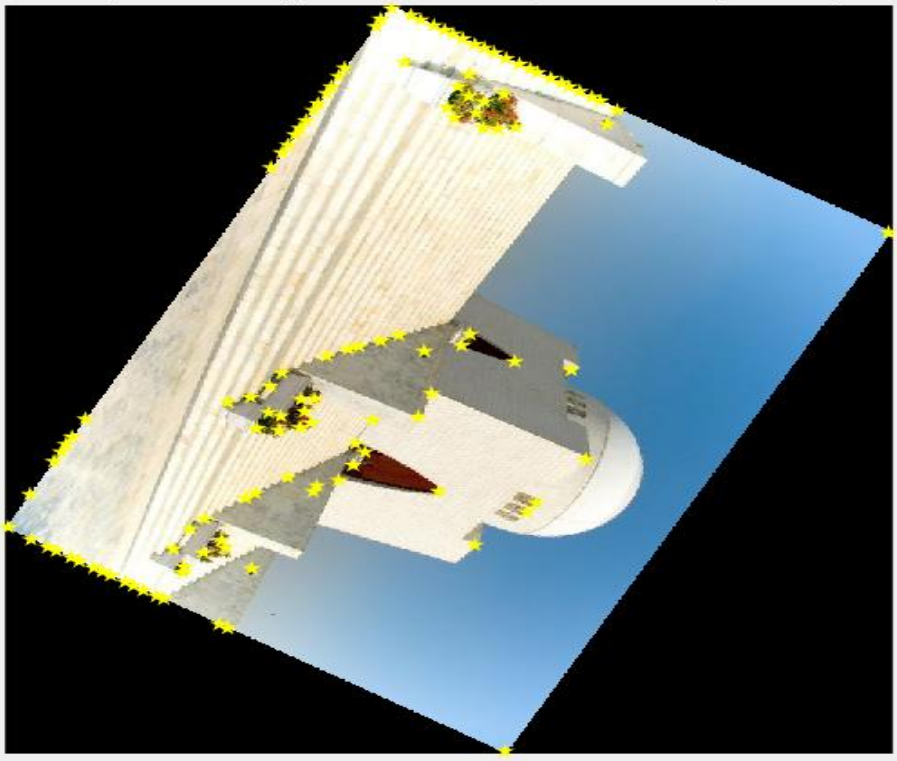
Rotated image with 210 degree where M = 71 , N = 71 and Repeatability M/N = 1



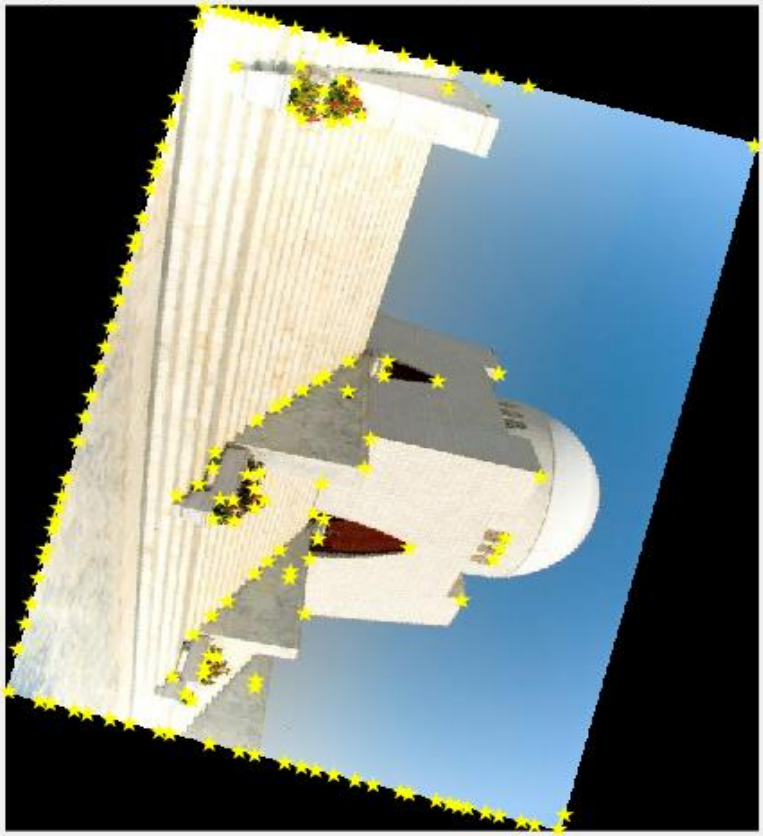
Rotated image with 225 degree where M = 71 , N = 71 and Repeatability M/N = 1



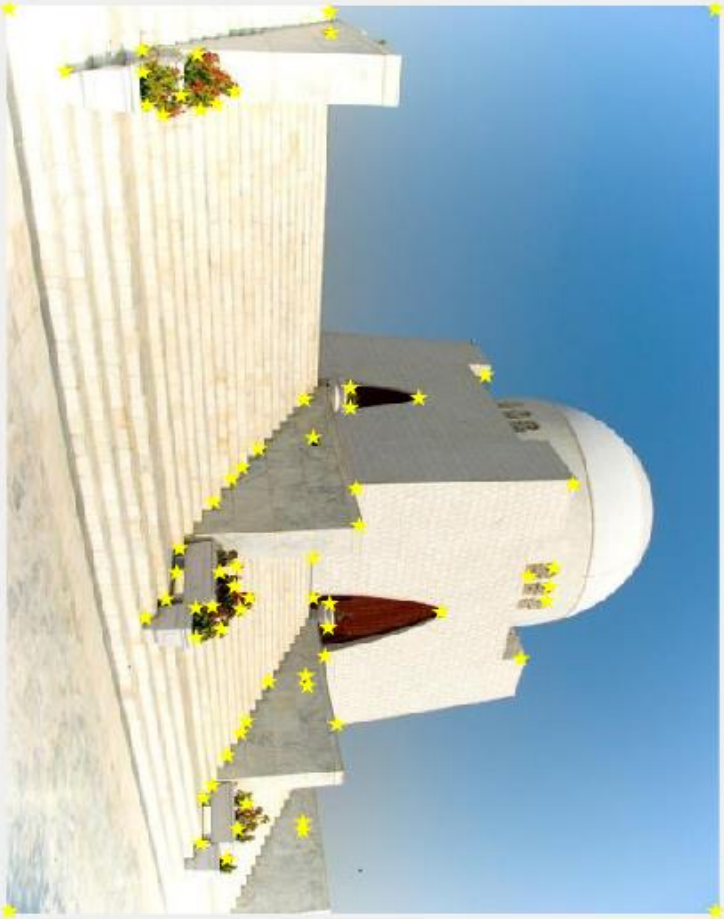
Rotated image with 240 degree where M = 71 , N = 71 and Repeatability M/N = 1



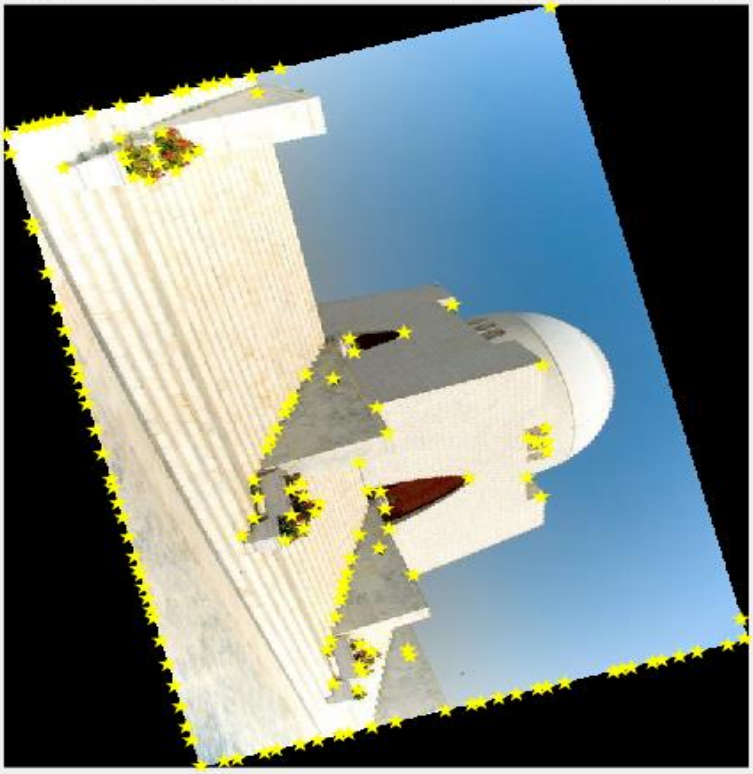
Rotated image with 255 degree where M = 71 , N = 71 and Repeatability M/N = 1



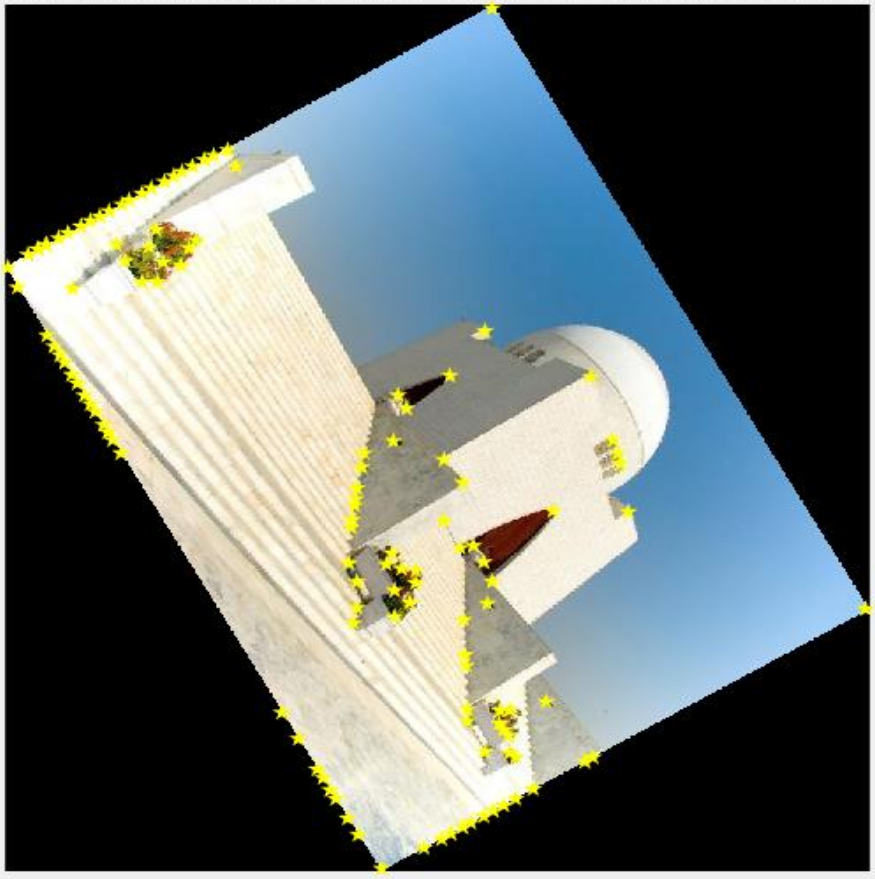
Rotated image with 270 degree where M = 71 , N = 71 and Repeatability M/N = 1



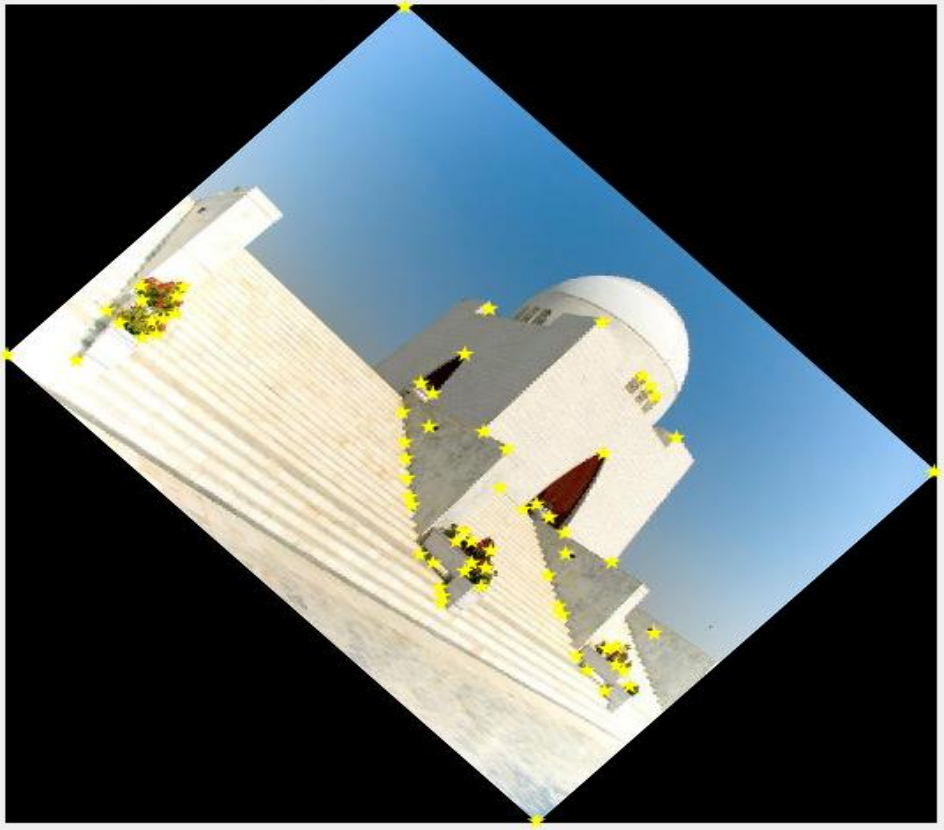
Rotated image with 285 degree where M = 71 , N = 71 and Repeatability M/N = 1



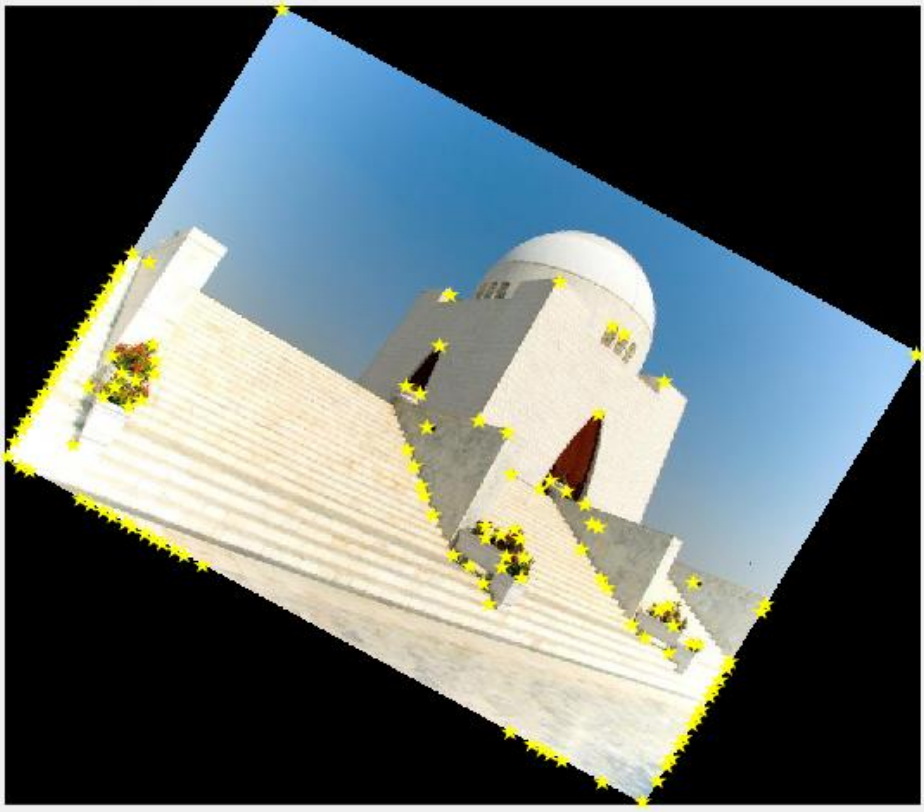
Rotated image with 300 degree where M = 71 , N = 71 and Repeatability M/N = 1



Rotated image with 315 degree where M = 71 , N = 71 and Repeatability M/N = 1

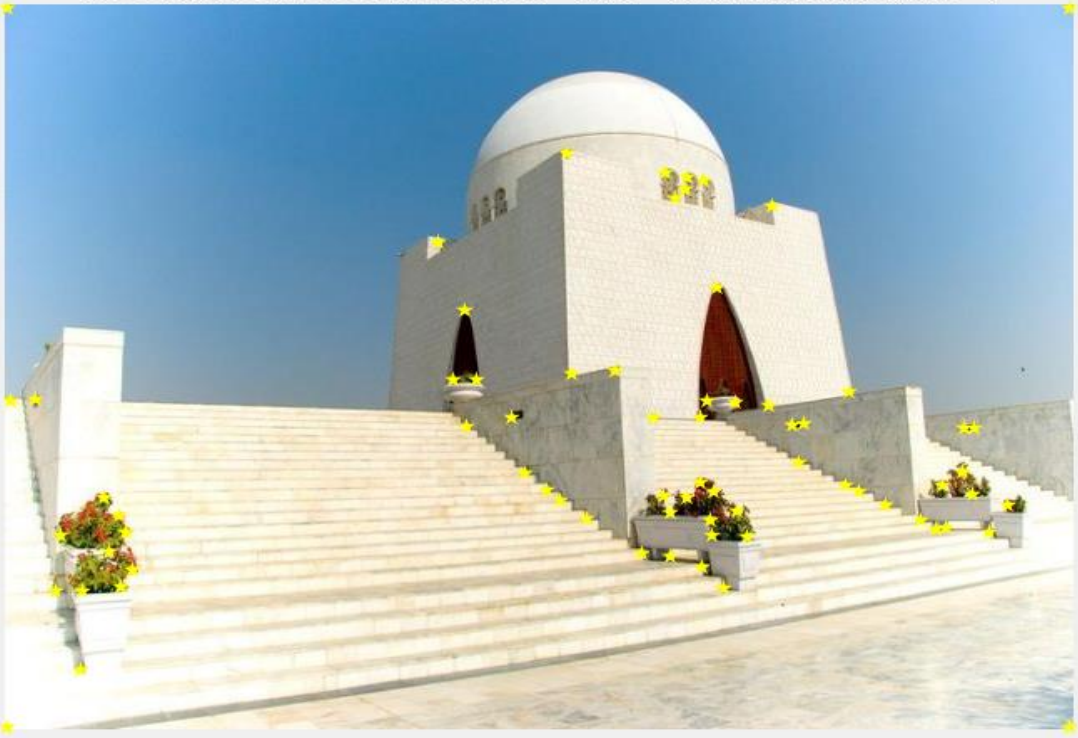


Rotated image with 330 degree where M = 71 , N = 71 and Repeatability M/N = 1

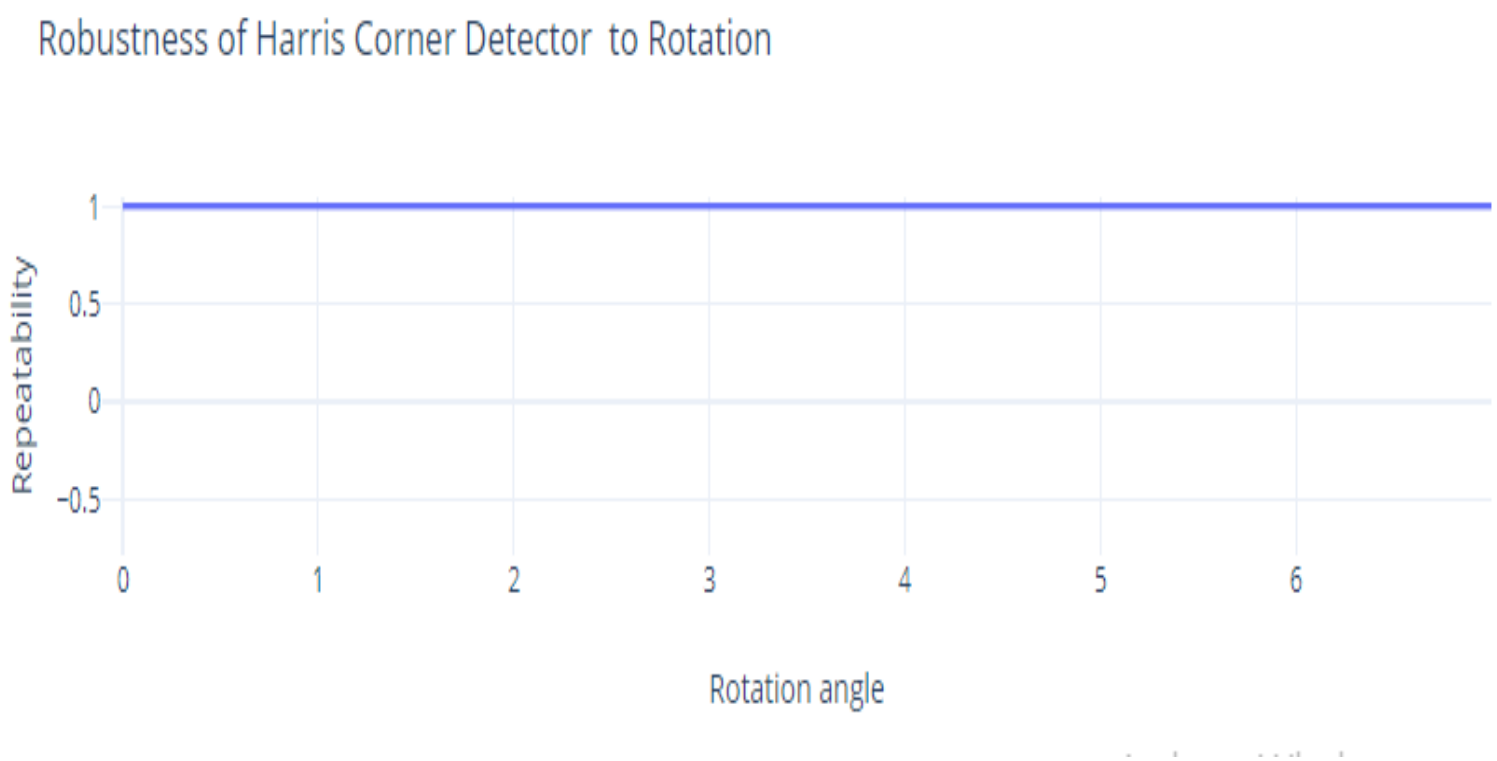


Rotated image with 345 degree where M = 71 , N = 71 and Repeatability M/N = 1

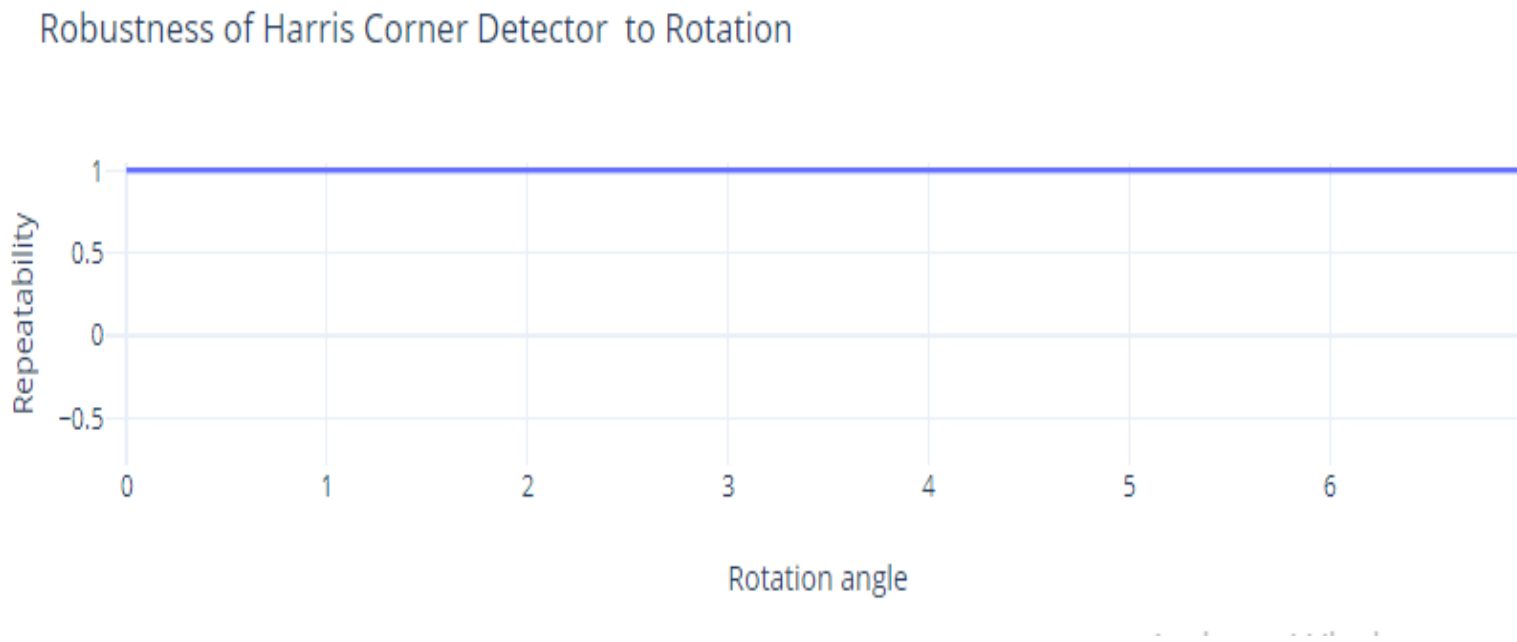
Rotated image with 360 degree where M = 71 , N = 71 and Repeatability M/N = 1



Repeatability vs Rotation Plot for the Image Famous Five



Repeatability vs Rotation Plot for the Image Mausoleum



Note: As a result of facing some error while plotting repeatabilty vs rotation, an online tool has been used . The link for online tool is given below.

https://chart-studio.plotly.com/create/

PART- C

Conduct an experiment analogous to part (b), but instead of rotating the image, resize the image by the scaling factors mˆ0, mˆ1, mˆ2, ..., mˆ8, where m = 1.2. Compute Harris keypoints for each resized image. You are allowed to use built-in library functions to perform this scaling, preferably with bicubic interpolation. By comparing Harris keypoints of the original and resized images, compute and plot repeatability against scaling factor. Comment on the Harris keypoint detector's robustness against scale changes.

The following built-in functions are used for computations:

- `sf=(1.2)^sfp, scaling factor for image`
- `Rimage1 = imresize(Oimagek,sf) for resizing image to different scales`

Code

```
1
2       % CS-867 COMPUTER VISION
3       % ASSIGNMENT-1,PART-C
4       % ROBUSTNESS OF HARRIS KEYPOINT DETECTOR TO SCALING
5
6 -     OimageI = imread('famous_five.png');
7 -     scale_eff(OimageI);
8       function scale_eff(Oimagek)
9 -     [Orows,Ocols] = harris(Oimagek);
10 -     Ocpoints = [Orows,Ocols];
11 -    for sfp=0:1:8  % RANGE OF SCALING FACTOR FROM 0 TO 8
12 -        sf=(1.2)^sfp; %SCALING FACTOR FOR IMAGE
13 -        Rimage1 = imresize(Oimagek,sf); %RESIZE IMAGE WITH SCALE FACTOR
14 -        [Rrows,Rcols] = harris(Rimage1);
15 -        Rcpoints = [Rrows,Rcols];
16 -        threshold = 4000;
17 -        Ocpointssize = length(Ocpoints);
18 -        N=length(Ocpoints);  % HARRIS CORNERS IN ORIGINAL IMAGE
19 -        rot_ang=15*pi/180; % ROTATION ANGLE FOR IMAGE
20          % TRANSFORMATION MATRIX FOR ROTATION OF KEYPOINT
21 -        RotMatrix = [cosd(rot_ang) -sind(rot_ang);sind(rot_ang) cosd(rot_ang)];
22          % RETURNS THE CENTER OF ORIGINAL IMAGE(GRAYSCALE)
23 -        CenterO = (size(rgb2gray(Oimagek))/2)';
24          % RETURNS THE CENTER OF ROTATED IMAGE(GRAYSCALE)
25 -        CenterR = (size(rgb2gray(Rimage1))/2)';
26 -        M=0; % COUNT OF MATCHED CORNERS
```

```
28
29          % PREDICTING IDEAL POSITION OF KEYPOINT
30 -      for i=1:1:Ocpointssize
31          % KEYPOINT ROTATED BY TRANSFORMATION MATRIX
32 -        RotatedP = RotMatrix*((Ocpoints(i))'-CenterO)+CenterR;
33          % EUCLIDEAN DISTANCE BTW TRANSFORMED & ROTATED IMAGE KEYPOINT
34 -        Euc_D = sqrt(sum((Rcpoints(i)-RotatedP).^2));
35 -        if(Euc_D<threshold) % IF DISTANCE LESS THAN THRESHOLD, THEN
36 -        M=M+1; % INCREMENT IN COUNT OF MATCHED KEYPOINTS
37 -        end
38 -      end
39        % REPEATABILITY MEASURE, RATIO OF MATCHED CORNERS TO ORIGINAL CORNERS
40 -      rep=M/N;
41 -      figure,imshow(Rimage1),hold on,
42 -          plot(Rcols,Rrows,'yp','MarkerFaceColor','y','MarkerSize',15),
43 -          title(['RESIZED IMAGE WITH m^',num2str(sfp),',  M =',num2str(M),
44          ' & N =',num2str(N),'  REPEATABILITY FACTOR,M/N =',num2str(rep)]);
45 -      end
46 -      end
47
48        % HARRIS KEYPOINT DETECTOR AS IN PART-A
49        function [R,C] = harris(image)
50 -      im = rgb2gray(image);  % CONVERTING GIVEN IMAGE FROM RGB TO GRAYSCALE
51 -      [dx,dy]=meshgrid(-1:1, -1:1);
52 -      ix = conv2(double(im),dx,'same'); % HORIZONTAL GRADIENT / VERTICAL EDGES
53 -      iy = conv2(double(im),dy,'same'); % VERTICAL GRADIENT / HORIZONTAL EDGES
54
55        % PARAMETERS FOR GAUSSIAN FILTER
56 -      sigma = 3;
57 -      radius=1;
58 -      order = (2*radius+1)^2;
59
60        % DEFINING GAUSSIAN FILTER
61 -      len = max(1,fix(6*sigma));
62 -      p=len; q=len;
63 -      [u1,u2]=meshgrid(-(p-1)/2:(p-1)/2, -(q-1)/2: (q-2)/2);
64 -      ug = exp(-(u1.^2+u2.^2)/(2*sigma^2));
65 -      [w,z] = size(ug);
66 -      sum = 0;
67 -      for i=1:w
68 -          for j=1:z
69 -              sum = sum+ug(i,j);
70 -          end
71 -      end
72 -      G = ug ./sum;
73
```

```
73
74      % COMPUTING ELEMENTS OF SECOND MOMENT MATRIX, M
75 -    Ix2 = conv2(double(ix.^2),G,'same');
76 -    Iy2 = conv2(double(iy.^2),G,'same');
77 -    Ixy = conv2(double(ix.*iy),G,'same');
78
79      % CORNERNESS MEASURE
80 -    r = (Ix2.*Iy2 - Ixy.^2)./(Ix2+Iy2 + eps);
81
82      % Value of Threshold set empirically
83 -    threshold = 4000;
84
85      % FINFDING MAX POINT FOR NON-MAX SUPPRESSSION
86 -    maximum_point = ordfilt2(r, order^2,ones(order));
87
88      % FINDING CORNERS
89 -    harris_corners = (r==maximum_point) & (r>threshold);
90 -    [R,C]=find(harris_corners);
91 -    end
92
```

Results

As we increase the scaling factor, the number of matched keypoints decreases. Hence, Harris detector is not robust to scaling.

RESIZED IMAGE WITH m^0, M =152 & N =152 REPEATABILITY FACTOR,M/N =1

RESIZED IMAGE WITH m^1, M =89 & N =152 REPEATABILITY FACTOR,M/N =0.58553

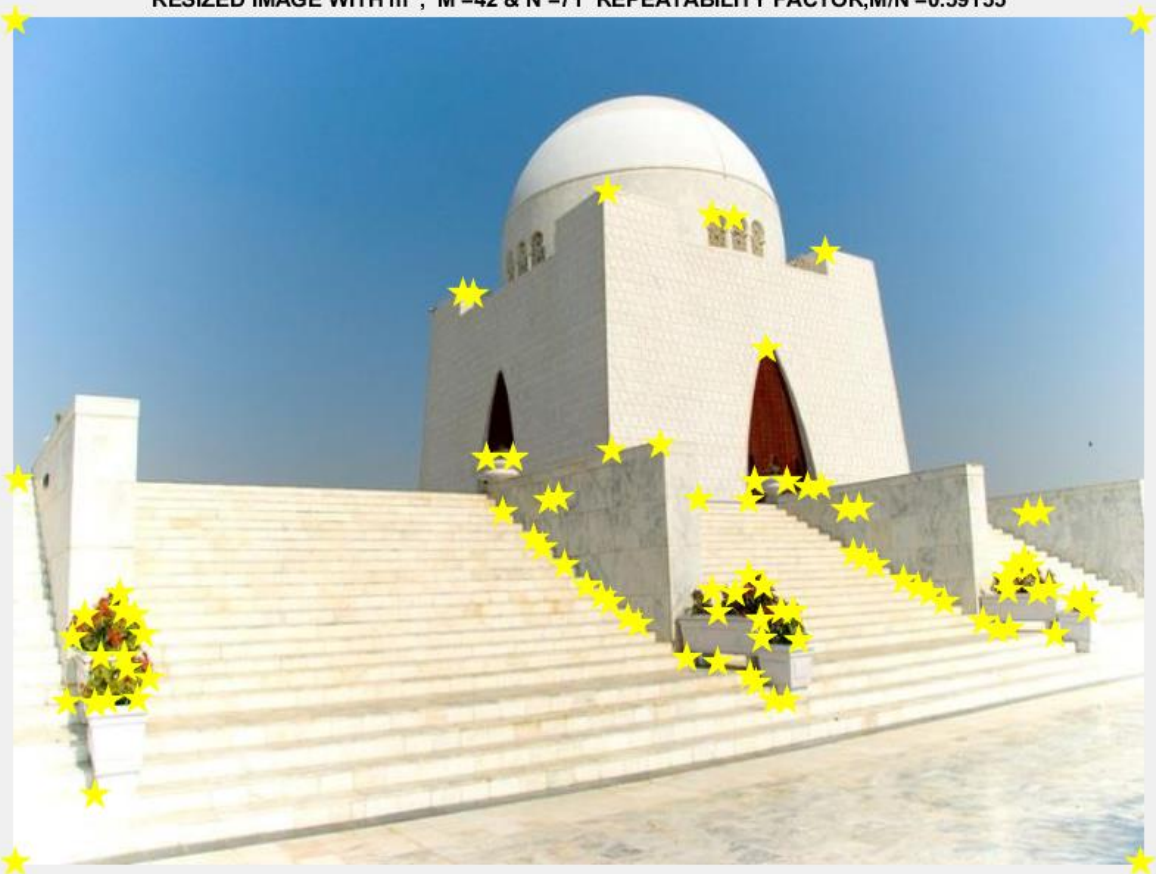
RESIZED IMAGE WITH m^2, M =36 & N =152 REPEATABILITY FACTOR, M/N =0.23684

RESIZED IMAGE WITH m^3, M =26 & N =152 REPEATABILITY FACTOR, M/N =0.17105

RESIZED IMAGE WITH m^4, M =24 & N =152 REPEATABILITY FACTOR,M/N =0.15789

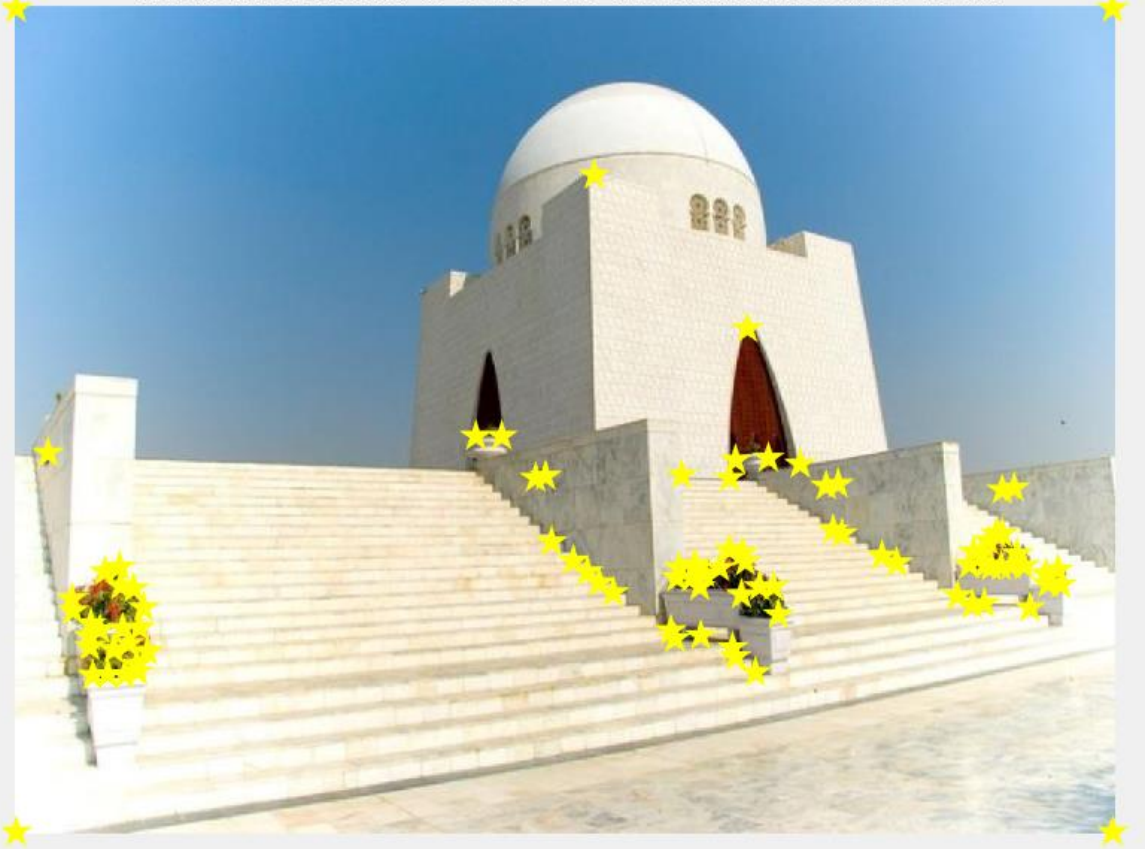
RESIZED IMAGE WITH m^0, M =71 & N =71 REPEATABILITY FACTOR,M/N =1

RESIZED IMAGE WITH m^1, M =42 & N =71 REPEATABILITY FACTOR,M/N =0.59155



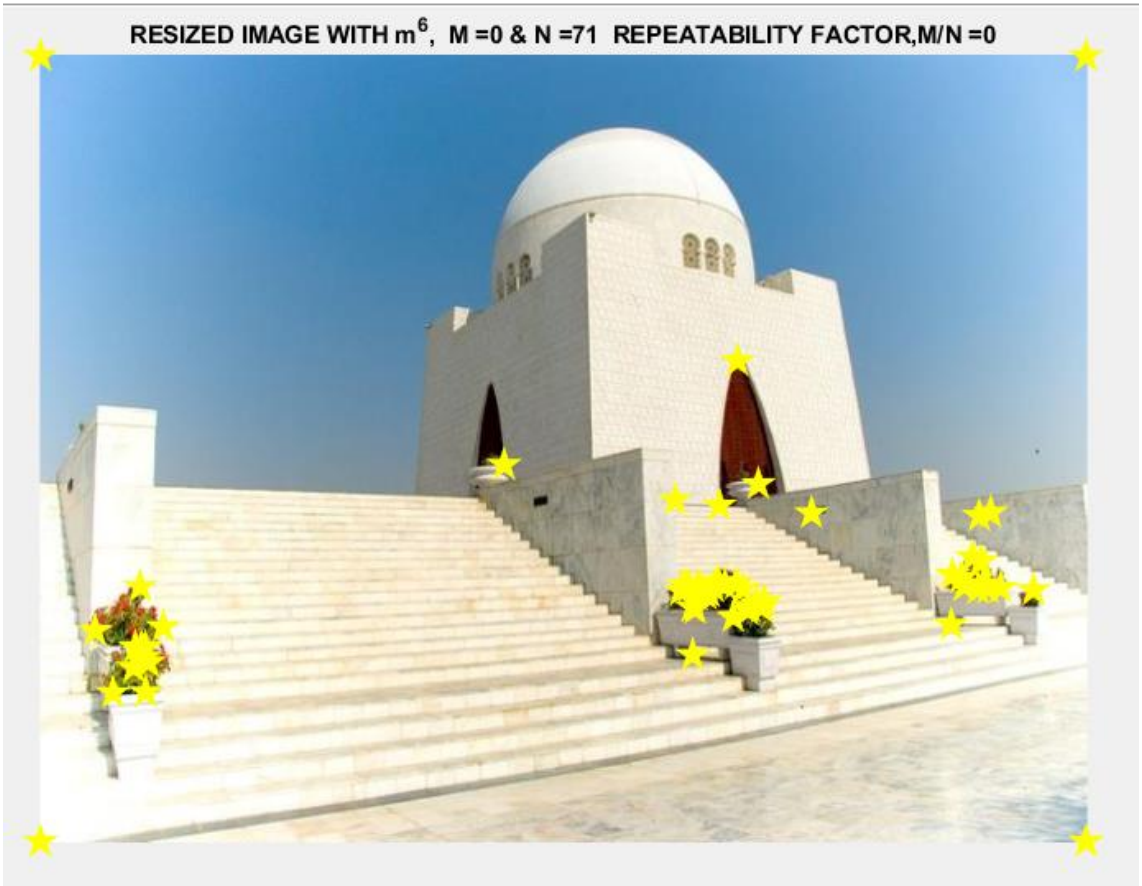
RESIZED IMAGE WITH m^2, M =32 & N =71 REPEATABILITY FACTOR,M/N =0.4507

RESIZED IMAGE WITH m^3, M =25 & N =71 REPEATABILITY FACTOR,M/N =0.35211



RESIZED IMAGE WITH m^4, M =13 & N =71 REPEATABILITY FACTOR,M/N =0.1831

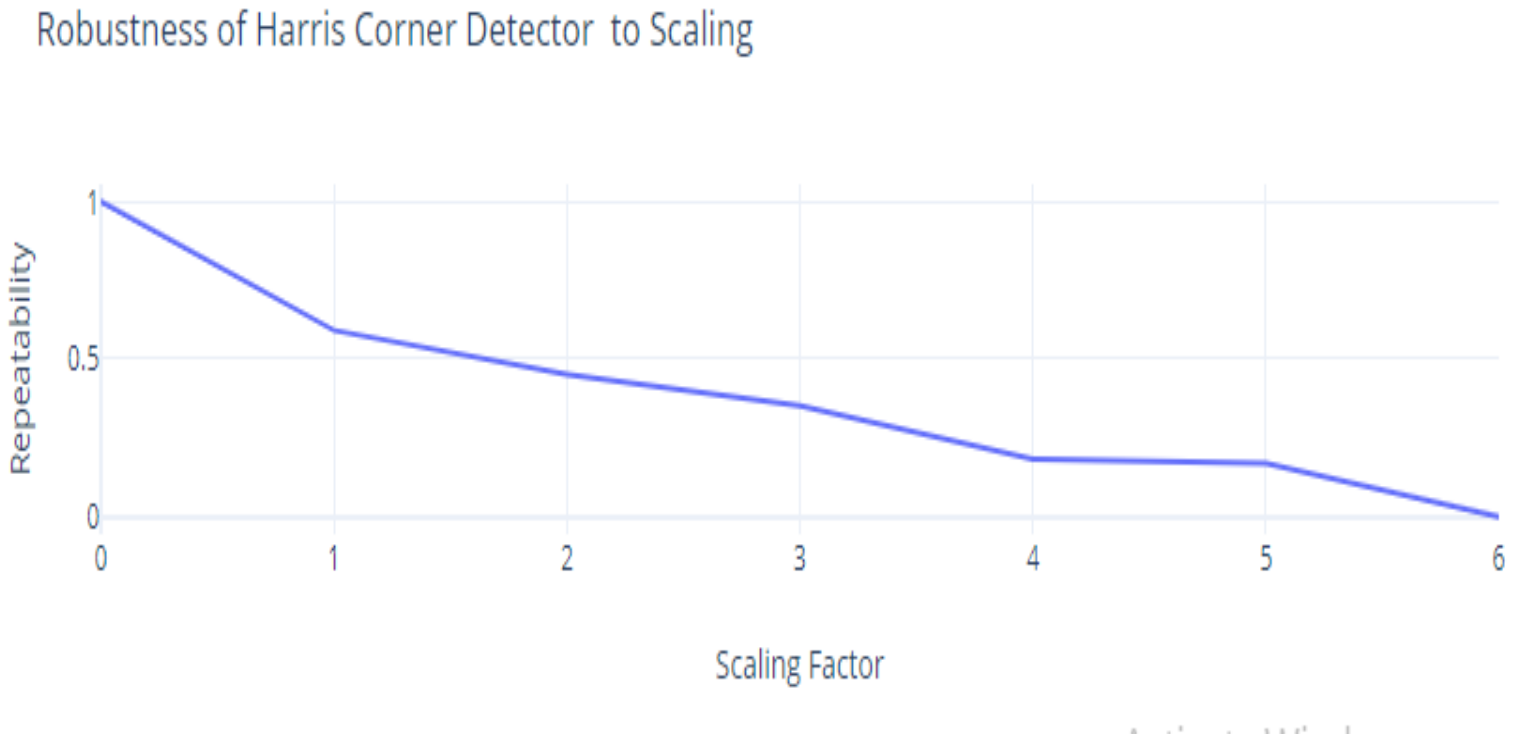
RESIZED IMAGE WITH m^5, M =12 & N =71 REPEATABILITY FACTOR,M/N =0.16901



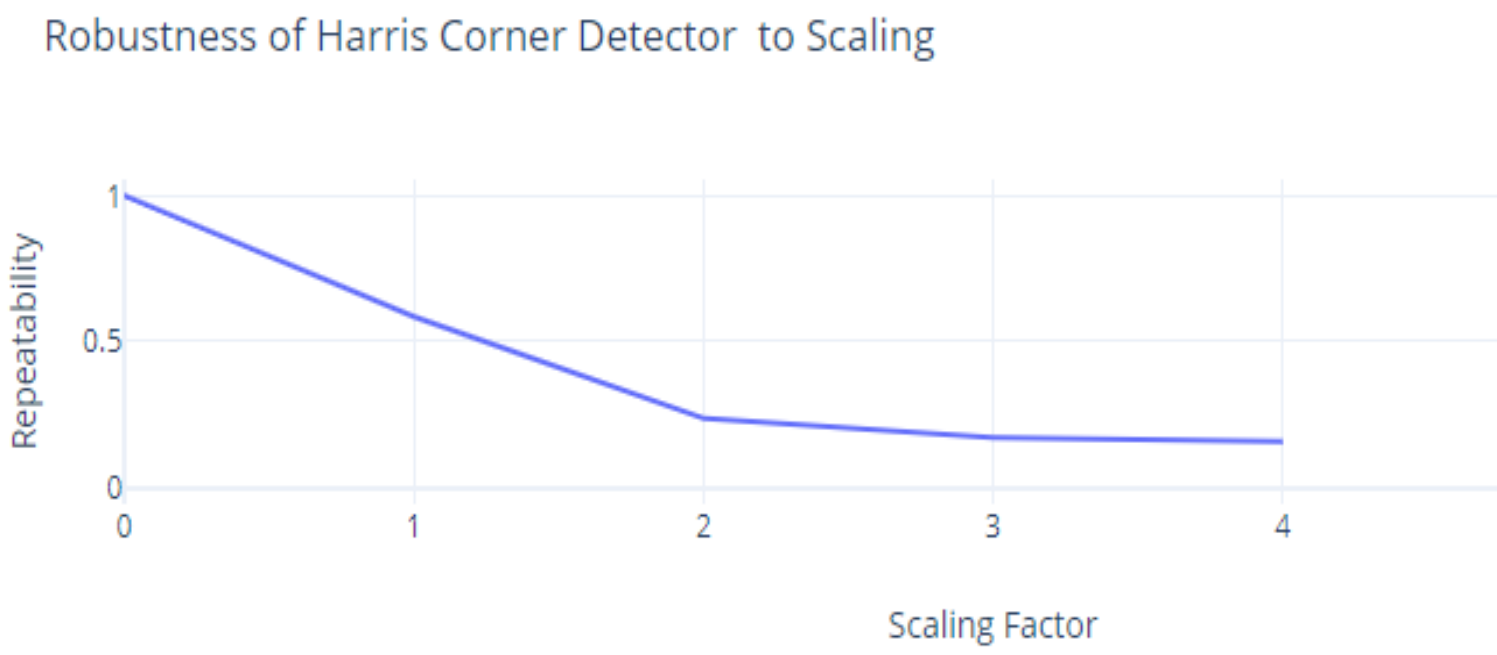
Note: As a result of facing some error while plotting repeatabilty vs scaling factor, an online tool has been used . The link for online tool is given below.

https://chart-studio.plotly.com/create/

Repeatability vs Scaling Factor Plot for the Image Mausoleum



Repeatability vs Scaling Factor Plot for the Image Famous Five



Conclusion

Repeatability plot for both images depicts that as we increase the scaling factor, the number of matched keypoints decreases which clearly shows that Harris corner detector is not robust to Scaling. In other words, Harris corner detector is not scale invariant.

Harris Corner Detector is invariant to Rotation but it is not invariant to Scaling. To some extent it has been shown in our results but not completely as it needs further work.

**********The End **********

****

****