Question 1

Gaussian Blur : kernel size = 7x7 | sigma = 3
This removes a lot of the inside the coins to prevent false circles. I had tried with 5X5 also but then in canny too many embeddings were coming.

Gray version

Gaussian Blur Version

Canny Edge on above image with :
Min = 45
Max = 80

Min = 48
Max = 87

As we are rotating theta from 0 to 360, this means that if there's not a significant change in cos(theta) and sin(theta), then we may get the same points as :
x = x_o + r*cos(theta)
y = y_o + r*sin(theta)

So if r*cos(theta) is small then we can have the same x and we'll be counting the x_o, y_o, r more times than required. So I've put a condition not to count if we get same x and y

I also put an additional constraint that not to count those x,y,r which have count < 3*r. As the total number of points is 2piR for a circle(it's circumference), approx 6R, so I'm taking circles which are than 3R.
Threshold was taken as maximum value of count in the hough matrix divided by 2.

Question 2

Used 12x12= 144 points to map corners from world to image.
In that I got 11 images which got mapped

Then I fed the points in world frame(objectpts) and corresponding imagepts to the Camera calibrate function.
This returned the intrinsic matrix, distortions, rotations and translations

Selected images :

"(0, ' ', True)
"(1, ' ', True)
"(2, ' ', False)
"(3, ' ', True)
"(4, ' ', True)
"(5, ' ', True)
"(6, ' ', True)
"(7, ' ', False)
"(8, ' ', True)
"(9, ' ', False)
"(10, ' ', False)
"(11, ' ', True)
"(12, ' ', True)
"(13, ' ', True)
"(14, ' ', True)

Fx = 538.11785671
Fy = 539.63623931
Principal shift in x = 327.92204497
Principal shift in y = 245.40485129

Distortion =[[-0.18805464 0.00155598 0.00735325 0.00049308 0.14002325]]

Below are the rotation x, y, z for the 11 images selected

Rvecs

```
[array([[-0.03442755],
        [ 0.09103046],
        [-0.01577683]]),
array([[-0.03634608],
        [ 0.10372135],
        [-0.00518211]]),
array([[-0.16120593],
        [ 0.42282827],
        [ 0.96600661]]),
array([[ 0.49218349],
        [-0.20811504],
        [ 1.34338358]]),
array([[ 0.46710466],
        [ 0.55725803],
        [ 1.50980899]]),
array([[ 0.39735132],
        [ 0.80377953],
        [ 1.5507719 ]]),
array([[ 0.0407269 ],
        [-0.81762122],
        [-0.02364989]]),
array([[-0.01734266],
        [ 0.31918912],
        [-0.03856701]]),
array([[-0.08044693],
        [ 0.28333866],
        [-0.04793964]]),
array([[-0.26769971],
        [ 0.48976805],
        [ 0.08026179]]),
array([[-0.36793967],
        [-0.36600749],
        [ 0.01767917]])]
```

Below are the translation x, y, z for the 11 images selected

tvecs

```
[array([[ -4.51265377],
        [ -5.31373073],
        [ 15.52207368]])
array([[ -4.88814011],
        [ -5.29552291],
        [ 16.38357896]])
```

```
array([[  2.49943704],
       [ -8.09106739],
       [ 22.55094776]])
array([[  7.32123078],
       [ -4.24610487],
       [ 17.15546934]])
array([[ 13.56949769],
       [ -3.42508939],
       [ 26.36761885]])
array([[  8.78198339],
       [ -0.1872941 ],
       [ 34.38038082]])
array([[ -1.6261897 ],
       [ -5.63373466],
       [ 17.19888245]])
array([[ -3.12477752],
       [ -5.68479638],
       [ 18.70546076]])
array([[ -1.23831447],
       [  0.24308372],
       [ 42.04310101]]),
array([[ -1.07359636],
       [ -3.83975669],
       [ 30.53319669]]),
array([[-10.27674794],
       [  3.87979902],
       [ 55.01803999]])]
```

Mean Error : 0.150199301248

Projection Error: [0.03846703 0.02730226 0.06302595 0.0308582   0.03226222 0.33431143
  0.29122409 0.1098037   0.3036073   0.05005315 0.37127699]

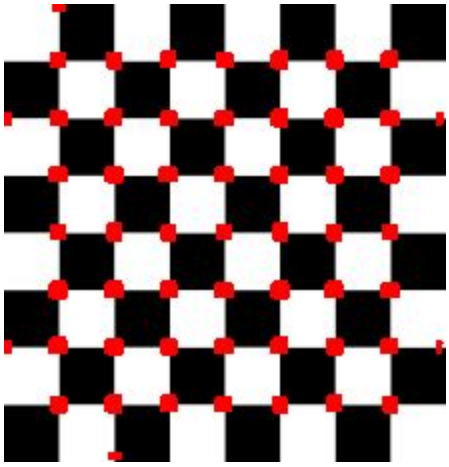
Source: https://docs.opencv.org/3.4.3/dc/dbb/tutorial_py_calibration.html

Question 3:

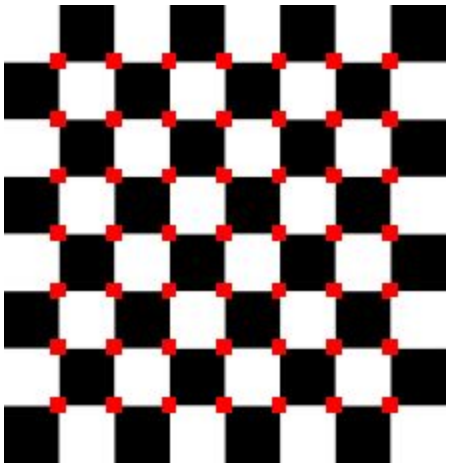
Threshold values(scale log10) :

CHESS :

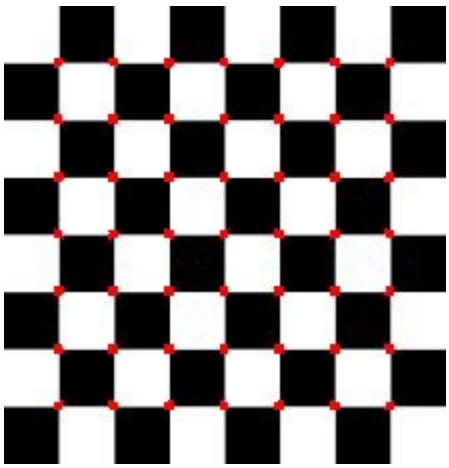
1. -2



2. 2

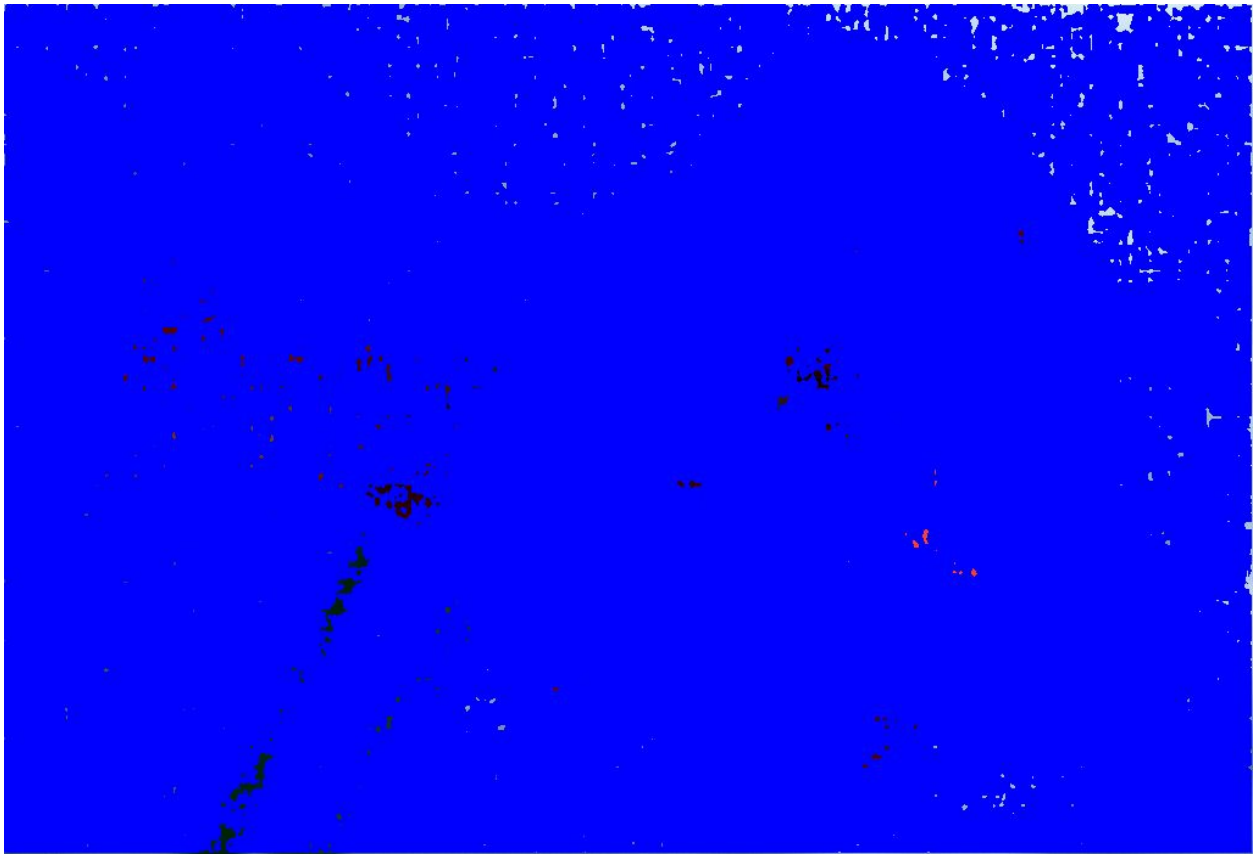


3. 4

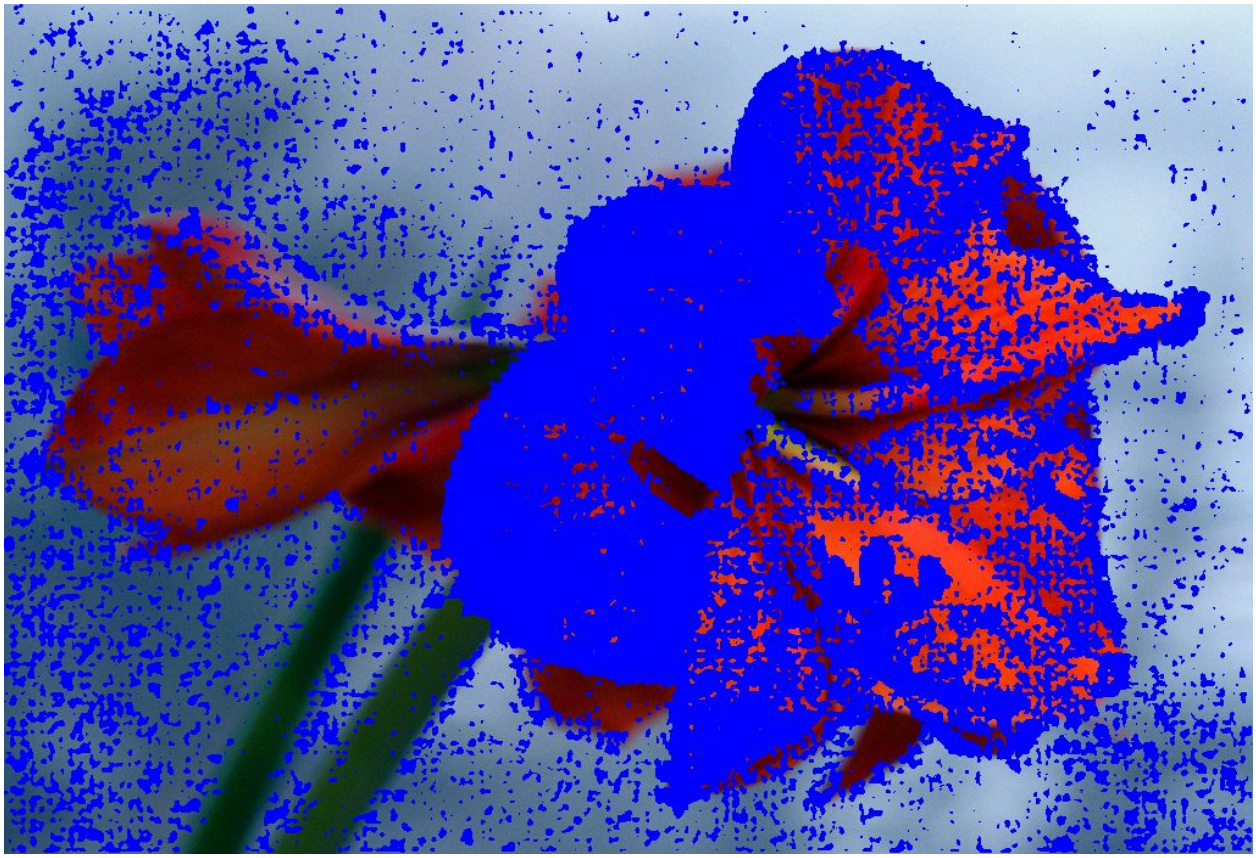


FLOWER:

1. 1 => 0



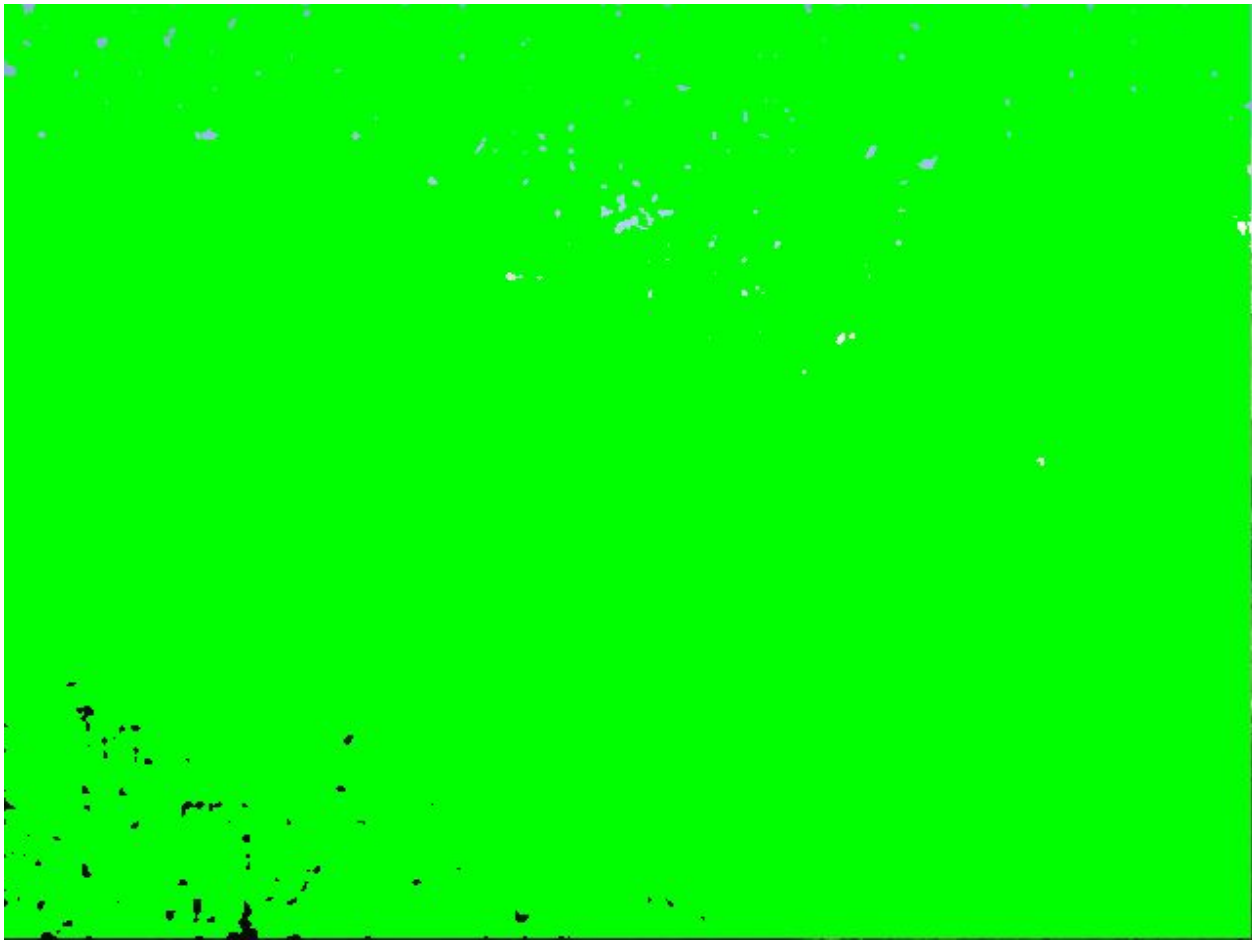
2. 10 => 1



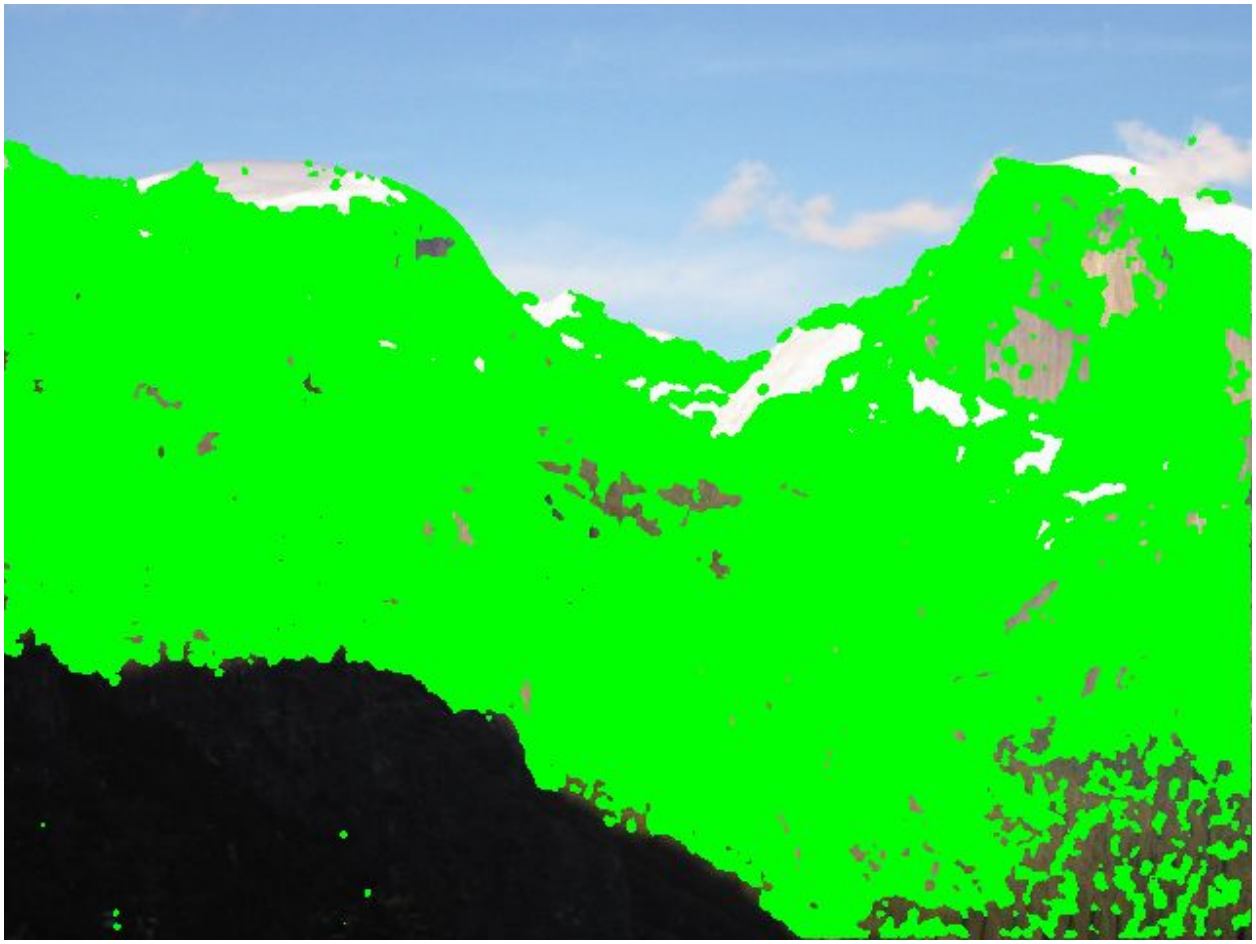
3. 100 => 2

YOSEMITE :

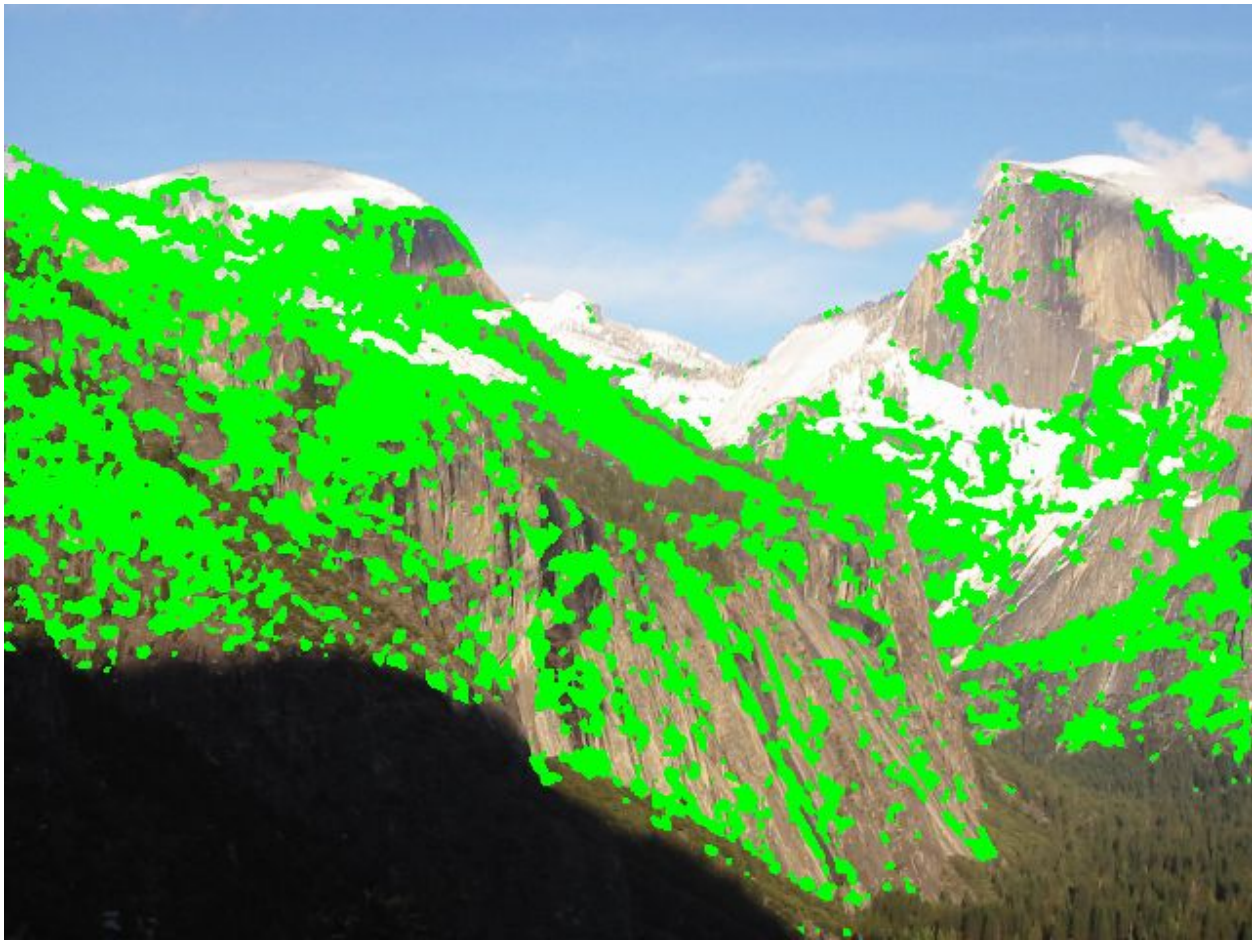
1. 1



2. 100



3. 1000

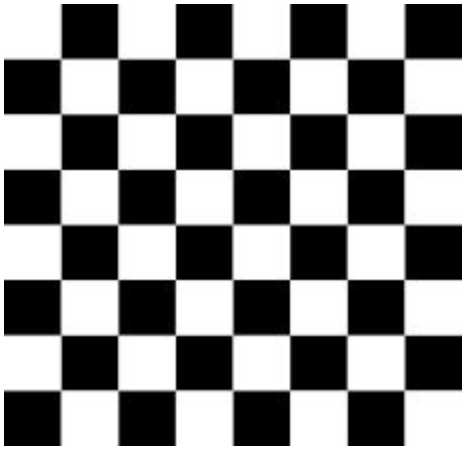


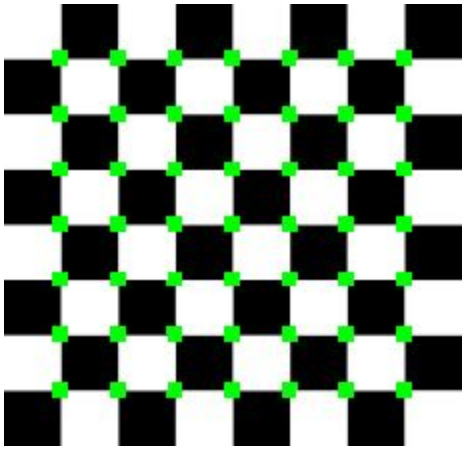
As we can see from the above experiments that increasing the threshold will reduce the number of corners having values less than the threshold removed. The threshold was put on eigenvalues. Same threshold for both the eigenvalues

ROTATION CLOCKWISE 90

Threshold = 100

Rotating the chess image has no effect on detecting corners, as seen below





COMPRESSED BY 2

Threshold = 100

Compressing the image and then taking corners doesn't have effect when threshold is high, even for threshold is 1, but for 10^(-2), the corners for compressed doesn't have additional corners towards the edges, but that may due to the grey color present between the white and black color