CV Assignment2

Kshitiz Jain (16051)

February 2019

1 Question 1

For detecting circles in a image using hough transform following steps were followed.

1.1 Downsampling

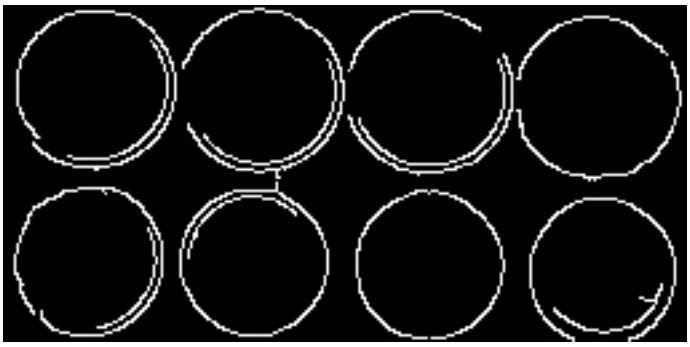
Since the photo was very large in size and we need to detect only distinct (majorly visible) circles, we downsampled the image by factor of 4, so as it took less time and only major information remains.

1.2 Blurring

The above obtained image still contains unnecessary high frequency information, hence Gaussian blurring is done with kernel size 3 and standard deviation 1.5.

1.3 Canny edge detection

Canny edge detection library of OpenCV was used to find edges of the above image. Canny edge detection involves thresholding in which the edge pixel values are set to 255.



1.4 Hough Transform

For every edge pixel there can be a circle, hence taking that pixel coordinate as centre in hough space do voting by constructing circles with varying radius. Now we select the most voted (Thresholding) coordinates in hough space and draw a circle in the original image by taking that coordinates and radius.

2 Question 2

The chessboard images given in the pdf are used for estimating the intrinsic and extrinsic parameters of the camera. Since there are multiple images, each image has it's own external matrix whereas there will give a single intrinsic matrix for the camera.
We use "cv2.findChessboardCorners" API for locating the chessboard corners, while the chessboard was actually 13x13 we locate 12x12 chessboard in the images so as to reduces the error in locating the board.
The above extracted image and object points we give to the calibrate API "cv2.calibrateCamera" and extract all the extrinsic and intrinsic parameters.
The results are as follows :

2.1 Intrinsic Parameters

- Focal Length : Fx = 538.1178 and Fy = 539.6362
- Skew constant : 0
- Principal Point : Px = 327.9220 and Py = 245.4048

2.2 Extrinsic Parameters

2.2.1 Rotation matrices

The rotation vectors obtained above was converted to Rotational matrices by using "cv2.Rodrigues" API.

Image	Rotation Matrix
Left1	$\begin{pmatrix} 0.99573573 & 0.01418558 & 0.09115442 \\ -0.017317 & 0.9992835 & 0.03365429 \\ -0.0906117 & -0.0350893 & 0.99526793 \end{pmatrix}$
Left2	$\begin{pmatrix} 0.99461295 & 0.00328862 & 0.10360628 \\ -0.00705469 & 0.99932673 & 0.03600431 \\ -0.10341813 & -0.03654127 & 0.99396651 \end{pmatrix}$
Left4	$\begin{pmatrix} 0.49478634 & -0.82391258 & 0.27632326 \\ 0.76197362 & 0.56420911 & 0.31790608 \\ -0.41783092 & 0.05325545 & 0.90696261 \end{pmatrix}$
Left5	$\begin{pmatrix} 0.22614681 & -0.96482522 & 0.1340519 \\ 0.87903868 & 0.14284301 & -0.45484819 \\ 0.41970063 & 0.22069927 & 0.88042218 \end{pmatrix}$
Left6	$\begin{pmatrix} -0.01895633 & -0.79359318 & 0.60815337 \\ 0.9984009 & 0.01737502 & 0.05379352 \\ -0.05325685 & 0.6082006 & 0.79199479 \end{pmatrix}$
Left7	$\begin{pmatrix} -0.15877775 & -0.7234425 & 0.6718784 \\ 0.96605096 & 0.02663453 & 0.256975 \\ -0.2038018 & 0.68987068 & 0.69465335 \end{pmatrix}$
Left9	$\begin{pmatrix} 0.68375321 & 0.00535385 & -0.72969369 \\ -0.03683294 & 0.99895162 & -0.02718456 \\ 0.72878315 & 0.04546429 & 0.68323357 \end{pmatrix}$
Left12	$\begin{pmatrix} 0.94876042 & 0.03516012 & 0.31403414 \\ -0.04064804 & 0.99911361 & 0.01094246 \\ -0.31337104 & -0.02314665 & 0.94934863 \end{pmatrix}$
Left13	$\begin{pmatrix} 0.95901599 & 0.03591875 & 0.28106613 \\ -0.05854383 & 0.99564748 & 0.07251694 \\ -0.27723807 & -0.08599959 & 0.95694468 \end{pmatrix}$
Left14	$\begin{pmatrix} 0.8800717 & -0.13991192 & 0.45376035 \\ 0.01223873 & 0.96197137 & 0.27287598 \\ -0.47468307 & -0.23459697 & 0.84831612 \end{pmatrix}$
Left15	$\begin{pmatrix} 0.93435813 & 0.04893952 & -0.35295864 \\ 0.08273006 & 0.93366486 & 0.34846185 \\ 0.34659864 & -0.35478845 & 0.86832859 \end{pmatrix}$

2.2.2 Translational Vectors (Tx,Ty,Tz)

Left1 = (-4.51265377,-5.31373074,15.52207368)

Left2 = (-4.88814011,-5.29552291,16.38357896)

Left4 = (2.49943704,-8.0910674,22.55094776)

Left5 = (7.32123078,-4.24610488,17.15546934)

Left6 = (13.56949769,-3.4250894,26.36761884)

Left7 = (8.78198339,-0.18729411,34.38038082)

Left9 = (-1.6261897,-5.63373466,17.19888245)

Left12 = (-3.12477752,-5.68479639,18.70546076)

Left13 = (-1.23831447,0.24308371,42.04310101)

Left14 = (-1.07359636,-3.8397567,30.53319669)

Left15 = (-10.27674794,3.879799,55.01803999)

2.3 Distortion Coefficients

- -0.18805464
- 0.00155598
- 0.00735325
- 0.00049308
- 0.14002325

2.4 Re-projection error

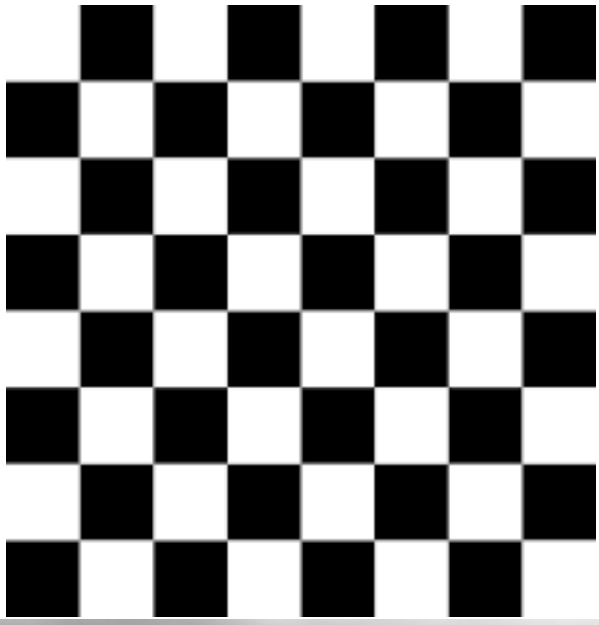
Re-projection for each image is given as followed :

- Left1 = 0.03846702661187315
- Left2 = 0.027302261323137404
- Left4 = 0.0630259471307111
- Left5 = 0.030858197469598933
- Left6 = 0.03226222469211951
- Left7 = 0.33431143018724696
- Left9 = 0.2912240883308437
- Left12 = 0.10980369652759304
- Left13 = 0.3036072968911244
- Left14 = 0.050053152497678555
- Left15 = 0.3712769920644868

Mean Re-projection Error : 0.1501993012478557

3 Question 3

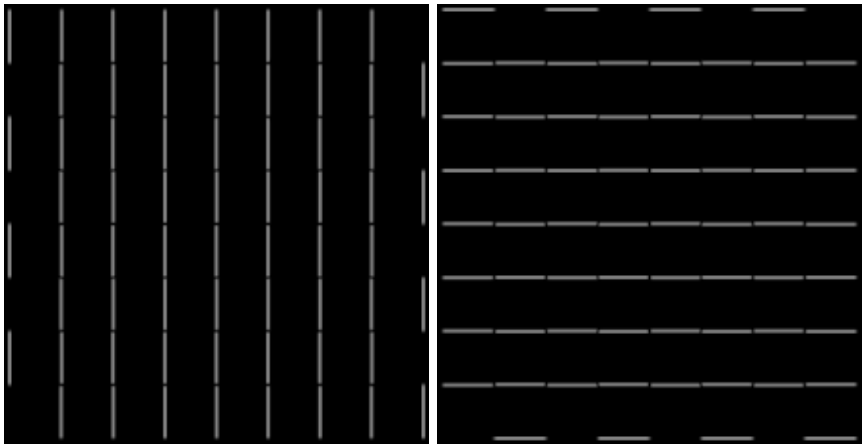
Corners are detected using Harris Corner Detection algorithm. The algorithm is applied on the following grayscale images.



3.1 Gradient of Images

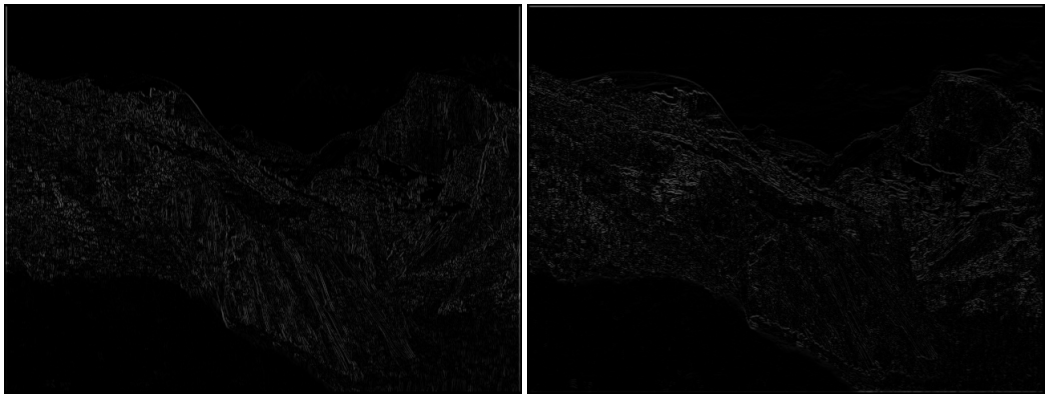
Gradients in both x and y directions of all three images were calculated using sobel filter of size 3x3.Results are as follows :

3.1.1 Chess



3.1.2 Flower

3.1.3 Yosemite



3.2 Covariance Matrix

Now for each element we calculate its covariance matrix using the elements within the 7x7 box where the element(whose matrix is calculated) is at the center. Matrix is calculated as follows :

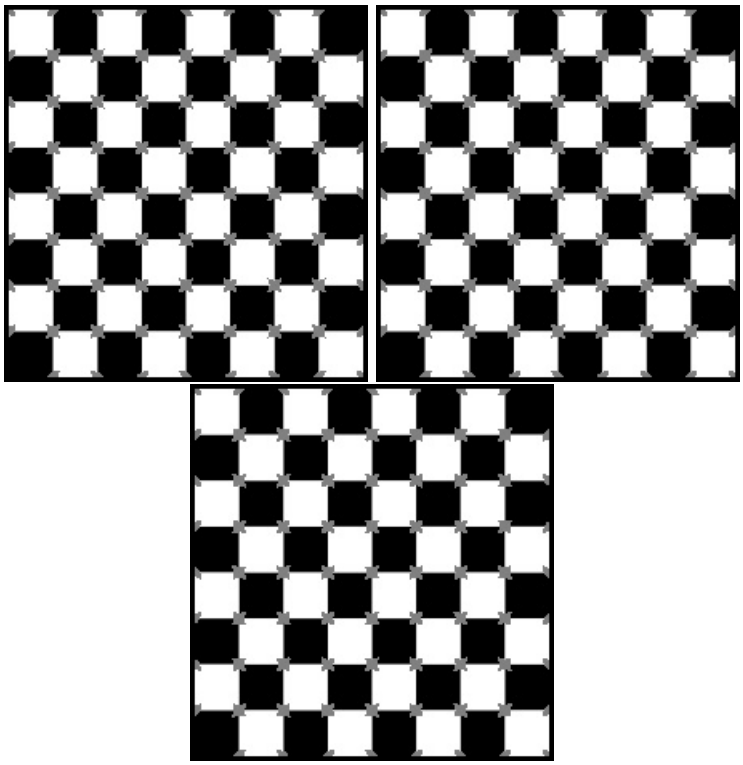
$$C = \begin{bmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{bmatrix} \qquad f_x = \frac{\partial f}{\partial x}, f_y = \frac{\partial f}{\partial y}$$

Now using the above matrix, its eigen values are calculated, now the trace and determinant of the eigen values are calculated and corners are now detected by comparing the response value with a threshold.

$$Response = Determinant - 0.06 * Trace^2$$

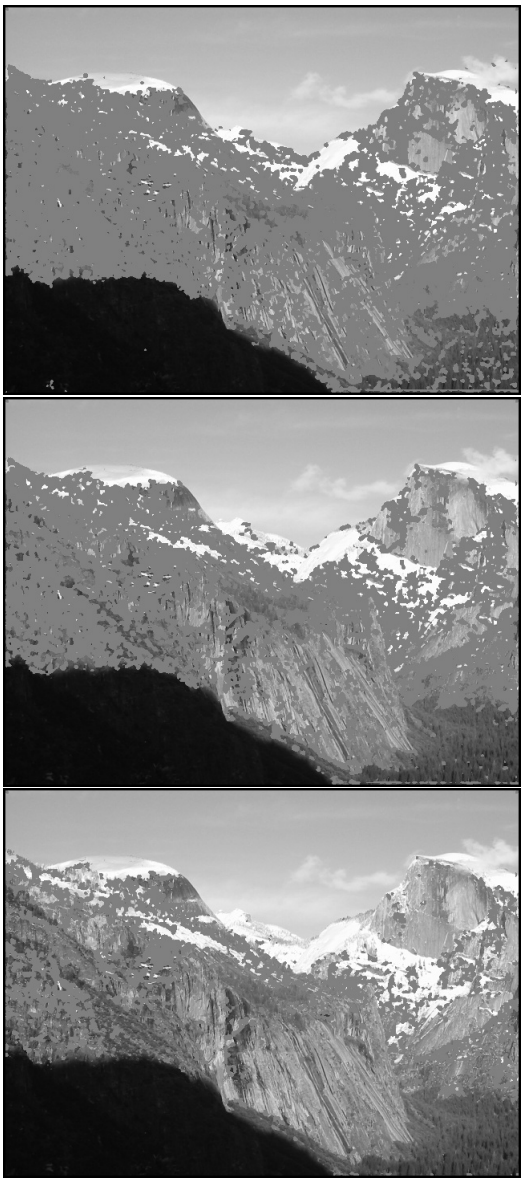
Corners are detected for 3 threshold values i.e. 100, 10000 and 100000 and respective results are as follows :

3.2.1 Chess



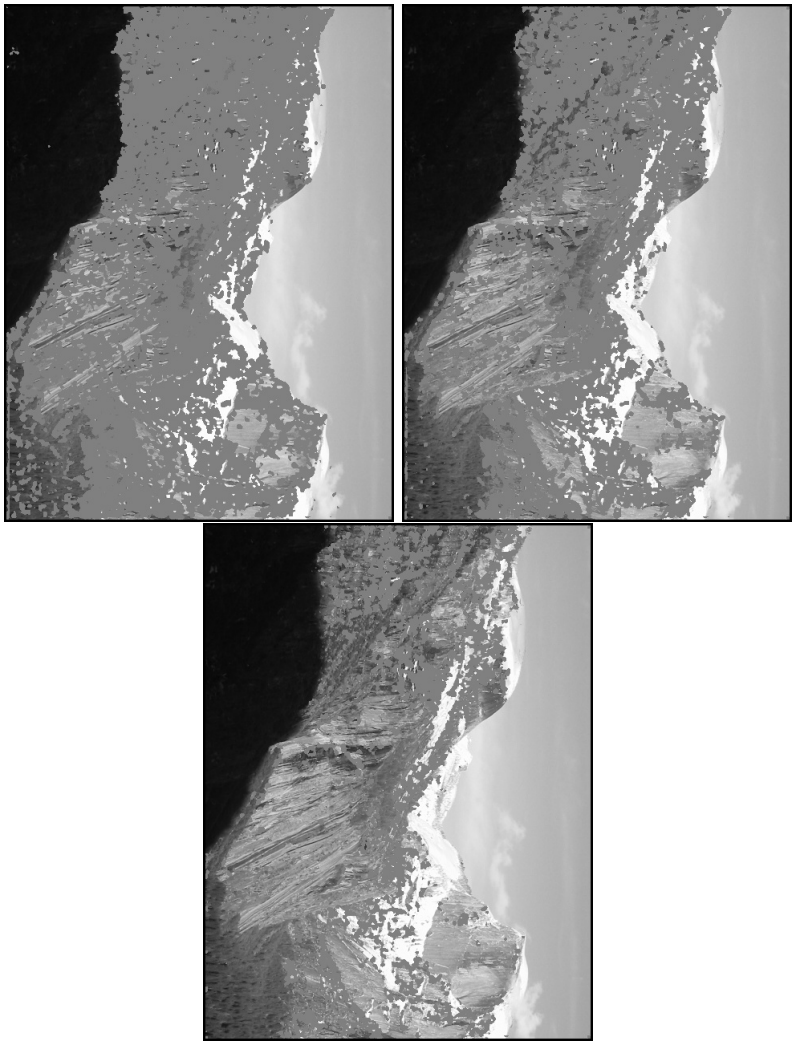
3.2.2 Flower

3.2.3 Yosemite



3.3 When images are rotated with 90 degrees

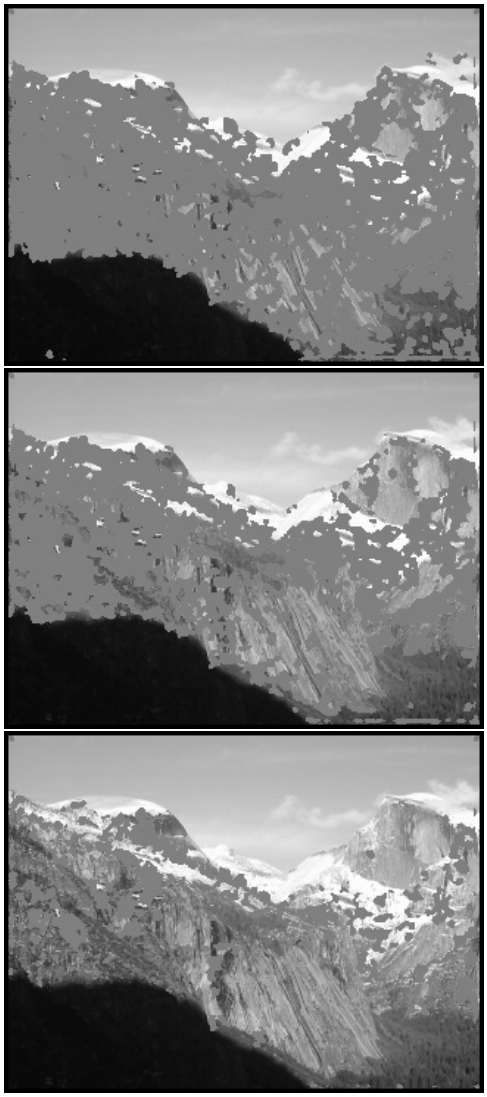
Yosemite image is used as the example. For rotating 90 degrees clockwise, we first take transpose of the image then do flipping(mirror image) of the image. Results are as follows :



Since the information of image and corners does not change in the rotated picture, there is no effect of rotating the image in 90 degree clockwise direction.

3.4 After Downsampling

Results are as follows :



We can see that only the prominent corners of the image retains in the downsampled image. New corners arises in downsampled image because when the image is downsampled the curve in the bigger image shrinks to a probable corner in the downsampled image. There are also some corners which do get get detected in the original image but not detected in the downsampled image, this is because loss of information due to downsampling.