# COMPUTER VISION
# ASSIGNMENT 1

Guru Pradeep Reddy (14CO246)

**Note : All the experiments are performed on building.jpeg file and as the images is little big zoom in to see the gradient images clearly.**

**1. Gaussian filter to reduce Noise**

Varied size of the Gaussian window and compared the results .

**3*3 window :**

Image file name : **gaussian_3*3.png**

**5*5 window :**

Image file name **: gaussian_5*5.png**

**Observation :**

As the original image itself looks clear without much noise application of filters just blurs the image a little the effect is little more prominent in case of 5*5 window size but as the image is of high resolution you can't exactly notice this difference until you zoom in. So in this case it won't affect the gradient much.

**2. Gradient and Direction calculation for Canny Edge detection**

Used Sobel operator for calculation of the gradient in both X and Y direction and the resultant direction is rounded of too 0, 45,90 and 135 respectively. Used sobel operator with 3*3 and 5*5 window for experiments

**Sobel 3*3 window Ix image :**

Image file name : **sobel_Ix_3*3.png**

**Sobel 3*3 window Iy image :**

Image file name **: sobel_Iy_3*3.png**

**Sobel 3*3 window gradient image :**

Image file name **: sobel_gradient_3*3.png**

**Sobel 5*5 window Ix image :**

Image file name : **sobel_Ix_5*5.png**

**Sobel 5*5 window Iy image :**

Image file name **: sobel_Iy_5*5.png**

**Sobel 5*5 window gradient image :**

Image file name **: sobel_gradient_5*5.png**

**Observations :**

- Ix gives the gradient in X direction and Iy will give gradient in Y direction. So they can be used to calculate edges in vertical and horizontal directions respectively because edges will be perpendicular to gradient direction.
- One more thing is in the sobel 5*5 window size gradients magnitude is more so it tends the edges look more bright.
- The outermost edges of the building are more clearly detected(high gradient value) compared to the edges on the building like near entrance. This is because in the outermost region right next to the boundary there is sky, so there will be clear difference in the intensity levels so it is easy to detect whereas in the inner parts there is not much difference near the interface so the gradient is relatively low there.

| **Sobel Window Size** | **Maximum gradient value** |
|---|---|
| 3*3 | 523.39 |
| 5*5 | 6831.48 |

**3. Non maximum Suppression :**

Here gradient of the pixel is compared with **two** pixels surrounding it depending upon the direction of gradient at that point this pixels may vary.
As we are considering the pixels which are local maximum in that neighborhood direction, the resulting image has thin edges as unwanted pixels which do not constitute the edge are removed. 5*5 sobel window size is capturing edges better than 3*3 one.
Image file names: **Suppressed_3*3.png , Suppressed_5*5.png**

**4. Hysteresis Thresholding :**

Tried with different parameters for minimum value of threshold and maximum value of threshold for 3*3 and 5*5 sobel window size images

**For 3*3 :**

Minimum value = 20 Maximum value = 50
Image file name : **thresholded_20_50_3*3.png**
Minimum value = 50 Maximum value = 100
Image file name : **thresholded_50_100_3*3.png**
Minimum value = 100 Maximum value = 200
Image file name **: thresholded_100_200_3*3.png**

**For 5*5 :**

Minimum value = 50 Maximum value = 100
Image file name : **thresholded_50_100_5*5.png**
Minimum value = 100 Maximum value = 200

Image file name : **thresholded_100_200_5*5.png**
Minimum value  = 300  Maximum value = 400
Image file name : **thresholded_300_400_5*5.png**
Minimum value = 600 Maximum value = 1000
Image file name **: thresholded_600_1000_5*5.png**

**Observation :**

- This two parameters are very important to get good edge image from this algorithm and there is no fixed value for these parameters which will work for all the images. It is entirely based on image and kernel sizes used for previous steps. So it has to be tuned carefully to get good result.
- For example here threshold 50 and 100 gives good edge image if we are using 3*3 window but it gives unnecessary edge if we use the same thresholds for 5*5 window. This because the range of magnitude of the gradients of 3*3 and 5*5 are different.

**5. HSV image**

This is edge image in which edges are colored based on the direction and magnitude of the gradient. Based on the direction different colors are assigned to it and based on magnitude of the gradient the intensity of the color is changed (Done by normalising the magnitude of the gradient in range 0-255)

Horizontal edges are given yellow color, vertical edges are given purple color while the edges in the diagonal directions are given pink and green colors (45 - Green, 135 - Pink). As experiments were performed on a building which doesn't have many curves most the edges are either in vertical or horizontal directions so yellow and purple colors dominate in the colored edge image. And as told before the outer edges have more gradient so they intensity of color will also be more there.
Image files : **hsvimage_3*3.png, hsvimage_5*5.png**

**6. Harris Corner detection.**

Two variations of windows are tried to find the better one. First one is matrix of all ones and second is matrix with gaussian weights. For window size 3, they will be like this

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

**Window with all ones**

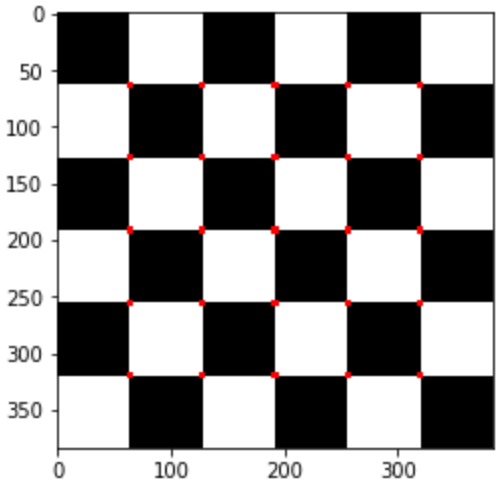
All the pixels in the neighborhood are given equal weight here.

| | | |
|---|---|---|
| **0.09329854** | **0.09329854** | **0.09329854** |
| **0.09329854** | **0.25361171** | **0.09329854** |
| **0.09329854** | **0.09329854** | **0.09329854** |

**Gaussian window of size 3 and standard deviation 1**

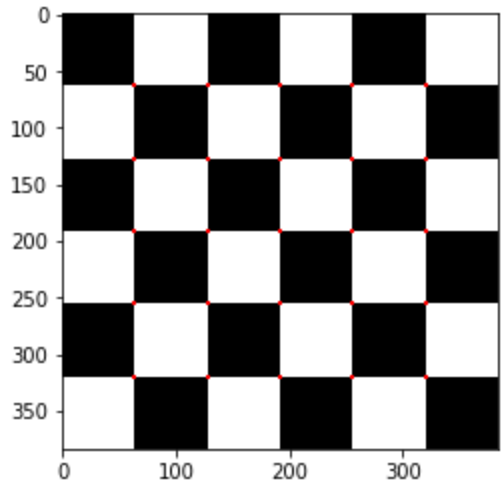
Experiments were conducted on chess board first because it easy to locate corners and easy to test which one performs better.

**With Normal window of size 3 :**




**No of corners : 900**

**With Gaussian window of size 3 :**




**No of corners** : 400

A threshold of 0.1*maximum_R_value of all pixels was used in both cases.
After this experiment it became clear that Gaussian window gives better result as it gives weights relatively thereby reducing the chance of pixel getting wrongly classified as corner.
Further experiments are performed using Gaussian window only.

Window size and threshold are varied and the results are compared.

**Window size 3 :**
Image file : **color_image_window_size3.png**
**Window size 5 :**
Image file : **color_image_window_size5.png**

As we increase the window size the no of corners that will get detected will also increase and no of wrongly detected corners will also increase. In this case 3*3 gives better results and fewer false corners than 5*5. But it depends upon the image and type of corner.

**Threshold (0.1 * Max R value) :**
Image file : **color_image_e1.png**

**Threshold(0.01 * Max R value)**

Image file : **color_image_window_size3.png**

**Threshold(0.001 * Max R value)**
Imagefile : **color_image_e3.png**

**Observations :**

- The original image is taken with camera of resolution 16Mp. So the obtained image is very big . So when it was used some corners are little stretched and are not detected properly, to overcome this problem i resized the image to little lower resolution using a online tool and then applied gaussian filter of 3*3 on it to compensate for the noise introduced while scaling. Corner detection was performed on this image. It gave better result and able to capture most of the corners in the image.
- Similar to the threshold parameter in canny edge detection it is also a very important to get the desired corners in the image and the best value depends upon the image and the application, if we want even small corners which are difficult to detect then the threshold should be low, If we want the corners only which are clearly distinguishable then we can set high value for threshold.
- Using Gaussian window gives better result than using window with all ones.