COMPUTER VISION : ASSIGNMENT 1

Name : Sandeep N Menon

Roll No : 14CO140

1. Canny Edge Detector

run ***python main.py <image_path>*** to test the code

The parameters analyzed are

a. Gaussian Kernel size

b. Sobel Kernel size

c. Hysteresis threshold (minVal,maxVal)

The output images are in the ***output*** folder with names of the form

type_of_output G=<Gaussian kernel size> S=<Sobel Kernel size> T=<(minVal,maxVal)>

a. Gaussian Kernel size

The Gaussian kernel sizes used in the experiment are **3x3** and **5x5**.

Images given in folder '**gaussian blur**'

Observation

There are no visible changes in the outputs of gradient image, canny output and edge colored image. This observation can be attributed to the fact that the original image has very less noise in it as it was taken using a high quality camera. Only with the presence of noise, can we find the canny detector misinterpreting noise as edges when there is less blurring.

b. Sobel kernel size

The Sobel kernel sizes used in the experiment are **3x3** and **5x5.** The direction obtained was rounded off to the nearest of the four angles 0,45,90 and 135.

Observation

Gradient Image

- The sobel **5x5 gradient images look brighter**. This is attributed to the high gradient magnitude calculated because of a larger sobel kernel.
- T**iny gradient differences are caught** in the higher sobel kernel as evident in the **sand from the bottom** of the picture and **sky at the top** of the picture.

- Gradients adjoining buildings and the sky or ground are more brighter because of the high change in intensity whereas edges seen within a building are relatively less brighter.
- The building is white in color. Hence in some areas it is evident that the sky and the building has very less difference. This is clearly seen in the gradient image near the **two water tanks.** The water tanks are black and hence show a high gradient difference whereas the roof to the left edge is white and show very less brightness in the gradient image.

Images given in folder **'gradient images'**

Suppressed Edge Image

Thick gradient lines are made thin by choosing only the gradient values which are higher along its direction.

- Sobel 5x5 had captured a lot of gradient information. Hence upon suppressing, there is still a lot of information. Even the slightest difference is captured and shown as edge.
- Sobel 3x3 shows better building structures as it is unpolluted with minor gradient changes within the image.

Images given in folder **'suppressed edge images'**

Hysteresis Thresholding

Initially the thresholds tried were (25,50), (50,100) and (100,150). These are thresholds on the gradient intensities.

Output images given in folder **' Canny detector output'.**

- As seen in output files of sobel 5x5 such as **cannyOutG=3_S=5_T=(25,50).png** and **cannyOutG=5_S=5_T=(50,100).png,** there are lot of unnecessary edge pixels which passes the threshold. This is attributed to the low threshold values. But only sobel 5x5 are affected by this. This is because of the high gradient values that sobel 5x5 calculates which was shown before in the bright gradient intensity images.
- To find better thresholds for varying sobel, the maximum gradient was found for both the sobel kernels.

Sobel Kernel Size	Maximum gradient value
3 x 3	459.05
5 x 5	6234.54

New thresholds were chosen as (100,200) and (200,250). Using these, the edges were under a better threshold as seen in the output in files **cannyOutG=3_S=5_T=(100,150).png** and **cannyOutG=3_S=5_T=(200,250).png**

- In conclusion, these threshold values depend on the gradient intensity values. In this problem statement, it depended directly on the size of the sobel operator. A threshold of (25,50) works if the sobel kernel is 3x3 (**cannyOutG=3_S=3_T=(25,50).png**), but gives lot of unnecessary pixels for the higher sobel kernel (**cannyOutG=3_S=5_T=(25,50).png**). In this case a higher threshold is required for a better result(**cannyOutG=3_S=5_T=(200,250).png**).

Edge Colored Map (using HSV model)

The output of the canny detector gives no information regarding the gradient intensity and orientation. To incorporate that, we color the canny edges based on their gradient intensity and orientation.

Using the HSV (Hue Saturation Value) model, we give different colors to edge pixels. For a higher gradient intensity we give a higher Value for a brighter edge, and for edges with different orientations we give different Hue values for different colors.

The gradient intensity is normalised to 0-255 as Value is 8-bit. The inclination of the edge was classified before to 0,45,90 and 135. We give the same angle for the Hue. This results in

Edge Inclination	**Color**
0 : Horizontal	Yellow
45 : Right diagonal	Green
90: Vertical	Blue
135: Left diagonal	Pink

Observations

- Normal building have mostly vertical and horizontal lines. Hence yellow and blue lines are mostly seen.
- Top left roof in the image shows Pink color as it is close to a left diagonal.
- The brightness of color based on the gradient intensity can be clearly seen between the outer edges of the building with same color as the sky and different color. The ones with almost the same color have a lower brightness(top right) compared to the ones with contrasting intensity(water tank).
- Images in folder **'Edge color map output'**

2. Harris Corner Detector

run ***python main.py <image_path>*** to test the code

The parameters analyzed are **window size and type.**

We experiment on 3x3 and 5x5 windows of normal window with all ones and Gaussian window. For a normal window all the pixels in the neighbourhood are given equal weights whereas a Gaussian window gives Gaussian weights to the neighbouring pixels.

A chess board image was used as a standard to check if the algorithm is identifying the easy corners.

For each window, the score is calculated using

R_score = det(window) - k*trace(window)^2

we set k=0.05

The pixel is considered as a corner if the R_score is greater than a threshold. This threshold is varied in our experiment to get different results. We take the threshold as a fraction of the maximum R_scoe value (**max_R_score)**

Observation

For chess board (**board.png)**

Window type	Window size	Number of corners	Output image name
Normal	3	1507	corner_marked_board_Normal_3.png
Normal	5	3153	corner_marked_board_Normal_5.png
Gaussian	3	1455	corner_marked_board_Gaussian_3.png
Gaussian	5	2134	corner_marked_board_Gaussian_5.png

- Clearly as window size increases, the number of corners detected also increases. But these are mostly false corners. So a 3x3 window size gives better results.
- Also, a Gaussian window detects less corners. The Gaussian removes redundant corner points around an actual corner thus giving better results. Hence a 3x3 Gaussian gives the best results

For **rbhavanCropped.png**

We take Gaussian window of size 3x3

Threshold	Number of corners	Output image name
0.5* **max_R_score**	26	corner_marked_rbhavanCropped_0.5R.png
0.1* **max_R_score**	324	corner_marked_rbhavanCropped_0.1R.png
0.05* **max_R_score**	608	corner_marked_rbhavanCropped_0.05R.png
0.01* **max_R_score**	1812	corner_marked_rbhavanCropped_0.01R.png
0.005***max_R_score**	2725	corner_marked_rbhavanCropped_0.005R.png

- Gaussian (3x3) blur was applied to the image to reduce noise.
- Experiments were conducted on the same image used for Canny edge detector but after cropping the areas with trees in it. This was done to avoid a large number of corners that are found in the leaves of the trees.
- The contrast of the image was increased using Histogram Equalisation to get better results. As the contrast increases, the gradient values across all directions from corners increase. Thus increasing R_score for corners.
- Using Gaussian window gives better result than using window with all ones.
- The threshold for the R_score depends on the number of corners we want to detect. The more corners we need, the lesser the threshold should be.