Intro to Image Understanding (CSC420)
Assignment 3

Due Date: Mar 7^{th}, 2025, 11:59:00 pm
Total: 160 marks

General Instructions:

- You are allowed to work directly with one other person to discuss the questions. However, you are still expected to write the solutions/code/report in your own words; i.e. no copying. If you choose to work with someone else, you must indicate this in your assignment submission. For example, on the first line of your report file (after your own name and information, and before starting your answer to Q1), you should have a sentence that says: *"In solving the questions in this assignment, I worked together with my classmate [name & student number]. I confirm that I have written the solutions/-code/report in my own words"*.
- Your submission should be in the form of an electronic report (PDF), with the answers to the specific questions (each question separately), and a presentation and discussion of your results. For this, please submit a file named **report.pdf** to MarkUs directly.
- Submit documented codes that you have written to generate your results separately. Please store all of those files in a folder called **assignment3**, zip the folder and then submit the file **assignment3.zip** to MarkUs. You should include a **README.txt** file (inside the folder) which details how to run the submitted codes.
- Do not worry if you realize you made a mistake after submitting your zip file; you can submit multiple times on MarkUs.

Part I: Theoretical Problems (50 marks)

[Question 1] Laplacian of Gaussian (25 marks)

The Laplacian of Gaussian operator is defined as:

$$\nabla^2 G(x, y, \sigma) = \frac{\partial^2 G(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 G(x, y, \sigma)}{\partial y^2} = \frac{1}{\pi\sigma^4}\left(\frac{x^2 + y^2}{2\sigma^2} - 1\right) e^{-\frac{x^2+y^2}{2\sigma^2}},$$

where the Gaussian filter G is:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The characteristic scale is defined as the scale that produces the peak value (minimum or maximum) of the Laplacian response.

1. **(10 marks)** What scale (i.e. what value of σ) maximises the magnitude of the response of the Laplacian filter to an image of a black circle with diameter D on a white background? Justify your answer.

2. **(5 marks)** What scale should we use if we want to instead detect a white circle of the same size on a black background?

3. **(10 marks)** Experimentally find the value of σ that maximizes the magnitude of the response for a black square of size 100×100 pixels on a sufficiently large white background.

   Hint: You can simply implement this task and automatically test for a large set of samples. You may also want to first generate the samples in log-domain to accurately locate the optimal value in a large spectrum.

1. **(10 marks)** What scale (i.e. what value of σ) maximises the magnitude of the response of the Laplacian filter to an image of a black circle with diameter D on a white background? Justify your answer.

In the 2-D Gaussian, we know that the scale, σ, acts as a sort of radius parameter as most of the distribution is within one σ. To maximize the response of the filter we want to maximize

$$\max (I * \nabla^2 G)(x, y, \sigma).$$

This will occur if the LoG is similar in size to the circle so that the edges are all captured in one correlation. As in the 'radius' of the LoG is similar to that of the circle. The LoG filter has the term

$$\left(\frac{x^2 + y^2}{2\sigma^2} - 1 \right),$$

which indicates that it changes sign at

$$x^2 + y^2 = 2\sigma^2.$$

The area around this will have a strong response in the convolution due to the differing signs, just like the Sobel filters. If we were able to match this to the size of the circle, this would capture the blob well.
We saw the LoG filter in lecture, it was increasing and positive and then flipped and crossed toward a negative local minima. This means, when corrleating the filter with a circle whose radius is the origin crossing, the area outisde the circle is sclaed positively, while the indiside is scaled negatively. For a black circle and white backgroud, the inside contribution is 0, and the outside contribution is maximized. Thus we desire a scale such that the LoG crossing point is the same as the border of the circle.
Continuing our derivation we get,

$$x^2 + y^2 = 2\sigma^2.$$
$$r^2 = 2\sigma^2$$
$$r = \sqrt{2}\sigma.$$

Using the relation $r = \frac{D}{2}$,

$$\sigma = \frac{D}{2\sqrt{2}},$$

which is the characteristic scale.

2. **(5 marks)** What scale should we use if we want to instead detect a white circle of the same size on a black background?

By the same reasoning as above, we would use

$$\sigma = \frac{D}{2\sqrt{2}}.$$

The only difference this time is that the circle is white and the background is black so the response will be negative. The absolute value of the response will be maximized which is the same as detecting it, but the value will be negative.

3. **(10 marks)** Experimentally find the value of σ that maximizes the magnitude of the response for a black square of size 100×100 pixels on a sufficiently large white background.

Using the code in "./assignment3/theory_1.3/experiment.py", I found the optimal σ is $\sigma = 0.10$. It seems that the code heavily favours lower values of σ.

[Question 2] Corner Detection (25 marks)

For corner detection, we defined the Second Moment Matrix as follows:

$$M = \sum_x \sum_y w(x,y) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

Let's denote the 2×2 matrix used in the equation by N; i.e.:

$$N = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

1. **(10 marks)** Compute the eigenvalues of N denoted by λ_1 and λ_2?
2. **(15 marks)** Prove that matrix M is positive semi-definite.

1. **(10 marks)** Compute the eigenvalues of N denoted by λ_1 and λ_2?

$$\det(N - \lambda I) = 0 \rightarrow \det \begin{bmatrix} I_x^2 - \lambda & I_x I_y \\ I_x I_y & I_y^2 - \lambda \end{bmatrix} = 0 \rightarrow (I_x^2 - \lambda)(I_y^2 - \lambda) - (I_x I_y)(I_x I_y) = 0$$

$$I_x^2 I_y^2 - \lambda I_x^2 - \lambda I_y^2 + \lambda^2 - I_x^2 I_y^2 = -\lambda I_x^2 - \lambda I_y^2 + \lambda^2 = 0$$

$$-I_x^2 - I_y^2 + \lambda = 0 \rightarrow \lambda = I_x^2 + I_y^2$$
$$\lambda_1 = 0 \qquad \lambda_2 = I_x^2 + I_y^2$$

As required.

2. **(15 marks)** Prove that matrix M is positive semi-definite.

The window function $w(x, y)$ is commonly a step function or a Gaussian, both of which are non-negative. To show that a matrix is PSD, we show that $x^T Ax \geq 0 \forall x \neq 0$. Since M is a weighted sum of matrix N, and the weights are non-negative, we can show that M is PSD by showing that N is PSD.

$$v^T N v = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} a \cdot I_x^2 + b \cdot I_x I_y \\ a \cdot I_x I_y + b \cdot I_y^2 \end{bmatrix}$$

$$= a^2 I_x^2 + 2ab I_x I_y + b^2 I_y^2$$

$$= (aI_x + bI_y)^2 \geq 0$$

Thus, we have shown that M is PSD.

Note that the proof for $v^T M v$ directly can be expressed as,

$$v^T M v = v^T \left(\sum_x \sum_y w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) v$$

$$= \sum_x \sum_y w(x, y) v^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} v$$

$$= \sum_x \sum_y w(x, y)(aI_x + bI_y)^2.$$

And since $w(x, y)$ is assumed to be greater than 0, our earlier simplification holds.

Part II: Implementation Tasks (110 marks)

[Question 3] Local Descriptor: Histogram of oriented gradients (70 marks)

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of scale-invariant feature transform (SIFT) descriptors, and shape contexts (a similar technique we have not seen in class), but differs in the sense that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. Until deep learning, HOG was one of the long-standing top representations for object detection.

In this assignment, you will implement a variant of HOG. Given an input image, your algorithm will compute the HOG feature and visualize it as shown in Figure 1 (the line directions are perpendicular to the gradient to show edge alignment).

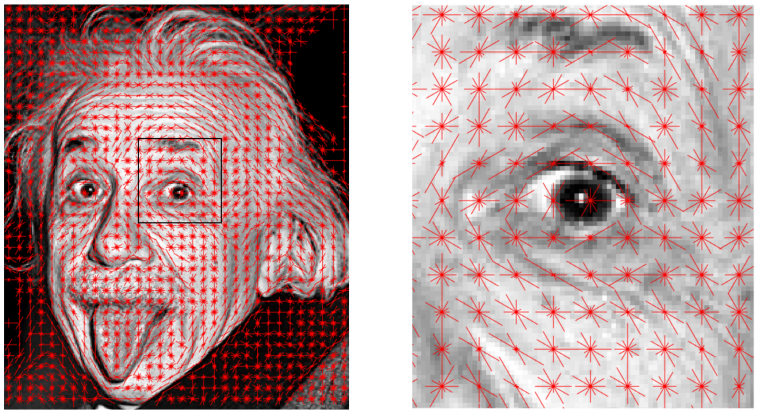


Figure 1: HOG features plotted on an example image.

The orientation and magnitude of the red lines represent the gradient components in a local cell. A HOG descriptor is formed at a specified image location as follows:

1. Compute image gradient magnitudes and directions over the whole image, thresholding small gradient magnitudes to zero. You should empirically set a reasonable value for the threshold for each of the input images.

2. Center a cell grid ($m \times n$) on the image. To create this grid cell, assume the grid cells are square and we have a fixed-size length for each of the cells in this grid; let us call that size τ. For example, if your image size is 1021×975 and $\tau = 8$, then you will have a grid size of $(m = 127) \times (n = 121)$. You can ignore the boundary of the image that can not be fit into a grid (on either end), i. e., just consider the crop of the image that fits to the grid perfectly, which is 1016×968 in this example.

3. For each cell, form an orientation histogram by quantizing the gradient directions and, for each such orientation bin, add the (thresholded) gradient magnitudes. This process

can be done in two steps: Imagine gradient orientations are discretized by 6 bins:

$$[-15°, 15°), [15°, 45°), [45°, 75°), [75°, 105°), [105°, 135°), [135°, 165°)$$

Remember 165° is equivalent to $-15°$ where the orientation is not directed. Now create a 3D array ($m \times n \times 6$) where in element (i, j, k) of this 3D array you will store the accumulated gradient magnitudes over all the pixels in the cell (i, j) with gradient orientations corresponding to bin k.
Another approach for constructing the HOG, is to collect the number of occurrences in each bin, rather than accumulating the magnitudes of occurrences; i.e. in element (i, j, k) of the histogram, we store the number of pixels in cell (i, j) with gradient orientations corresponding to bin k

(15 marks) Choose reasonable values for the threshold and cell size, and then visualize the resulting 3D arrays (using both approaches) on the given images similar to the quiver plot of Figure 1. Briefly, compare the two approaches by inspecting the visualizations.

Hint: You can use any package/function for creating the visualization in Figure 1. One way to do that is to superimpose 6 quiver plots (one for each bin), generated by *quiver* function in *matplotlib* package.

For the remaining tasks, you can use either approaches for constructing HOG. Make sure to explicitly mention your choice in the report.

4. To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks (adjacent cells). Given the histogram of oriented gradients, you apply L2 normalization as follows:

   - Build a descriptor for the first block by concatenating the HOG within the block. You can use block_size = 2, i.e., 2×2 block will contain $2 \times 2 \times 6$ entries that will be concatenated to form one long vector.
   - Normalize the descriptor as follows:

     $$\hat{h}_i = \frac{h_i}{\sqrt{\sum_i h_i^2 + e^2}}$$

     where h_i is the i^{th} element of the vector and $\hat{h}_i$ is the normalized histogram. e is the normalization constant to prevent division by zero (e.g., $e = 0.001$).
   - Assign the normalized histogram to the first cell of a new histogram array, i.e. cell (1,1).
   - Move to the next block of old histogram array with stride 1 and iterate steps 1-3 above, to compute the next cell of the new histogram array.

(15 marks) The resulting new histogram array will have the size of $(m-1)\times(n-1)\times 24$. Compute normalized histogram arrays for the provided images, and store them in a single line text file where the data is stored row by row (first row then second row etc.). Submit both your code and the files that are generated by your code. Please note that the file should have the same name as the image (e.g. 'image.jpg' $\rightarrow$ 'image.txt').

In addition to the provided images, use your own camera (e.g. smartphone camera) to capture two images of the same scene, one with flash and one without flash. Convert the images to gray-scale, and down-sample the images if needed to avoid excessive computation overhead.

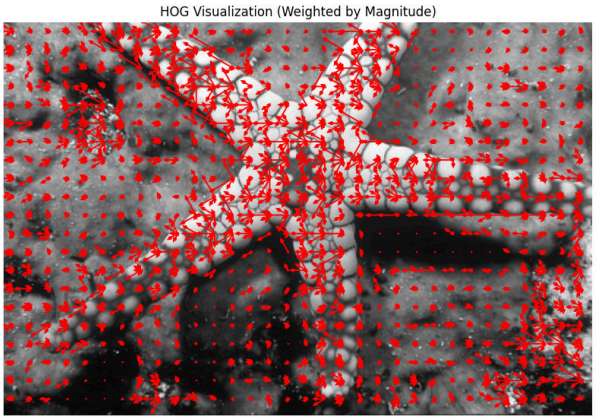
(10 marks) First, compute the original HOG arrays for these two images and visualize them similar to Figure 1.

(10 marks) Second, compute the normalized histogram arrays for each of these two images, and store them in two txt files as instructed earlier.

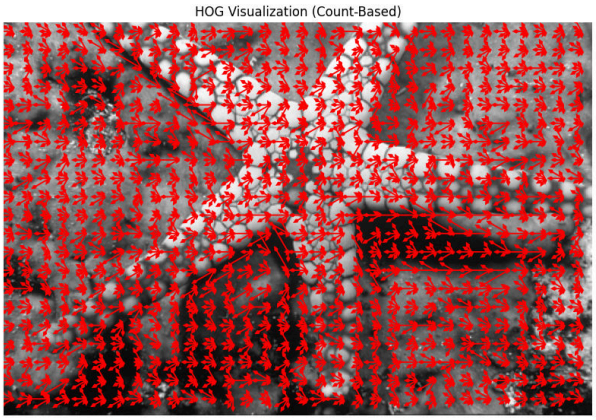
(20 marks) Third, by comparing the results, argue why or why not the normalization of HOG may be beneficial. Limit your discussion to a paragraph, containing the main points. You can compare the histograms visually or you may want to define a quantifiable measure to compare the histograms for pair of with-flash/no-flash images. If you choose to visually compare, provide the details of your visualization approach for normalized HOG; alternatively, if you decide to quantitatively compare the histograms, include the function you used and your justification in the report.

I went with the magnitude approach for part 4 and beyond. All pictures and text files can be found in "./assignment3/task1/assets/".

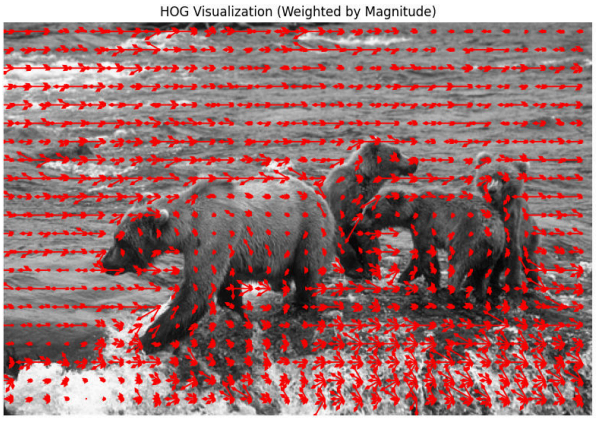
Note Figure 2 below demonstrates the count and magnitude approaches for both images provided. It is apparent that the magnitude approach better understands color differences and corresponding boundaries. The count-based HoG is rather noisy and picks up gradients everywhere as it focuses on count and catches very small gradients. It still is catching the same details, but just more so. It can be said, in a sense, that the magnitude based HoG is better able to distinguish meaningful gradients, while the counter-based HoG is better at getting all the gradients. Each may have its case. There are many regions on the magnitude HoG that are nearly zeroed out. These correspond to darker regions with low contrast as one would expect.
As the count based approach is not dependent on the magnititude, the threshold parameter has no impact on it.



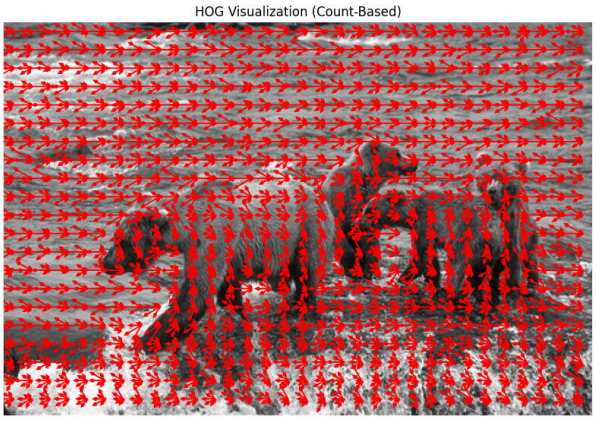
(a) Image 1 HoG using HoG Magnititude.



(b) Image 1 HoG using HoG Counts.



(c) Image 2 HoG using HoG Magnititude.



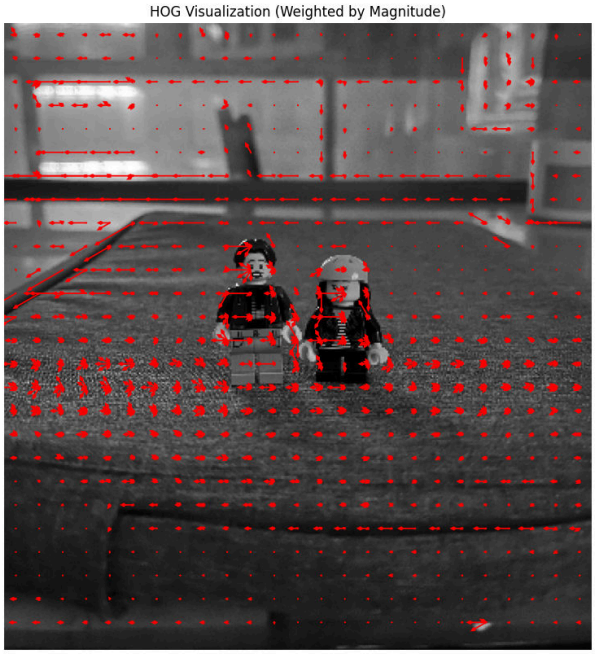
(d) Image 2 HoG using HoG Counts.

Figure 2: Combined plots of HoG using Magnitudes and Counts.

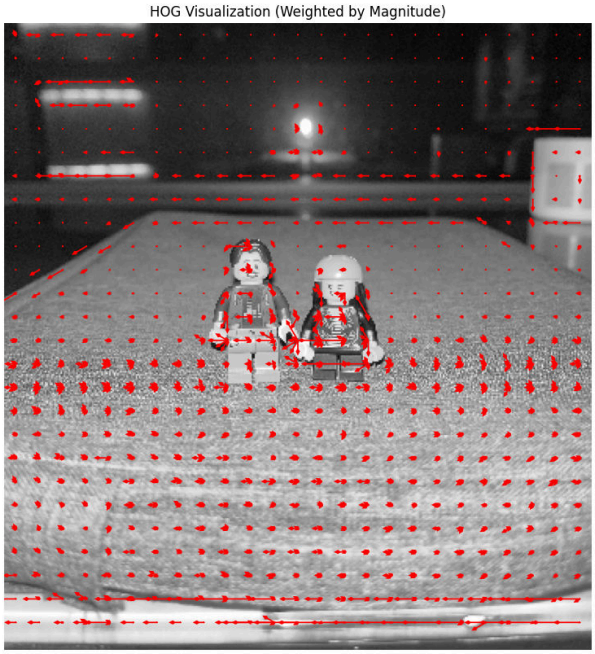
Note once again that I chose to proceed with the magnitude-based approach. The next two parts to compute the original and normalized HoG arrays for the flash/no-flash images and visualize them. These plots are provided in Figure 3.

Figures 3(a) and 3(b) reveal that the original (non-normalized) magnitude-based HOG show differences between the flash and no-flash images. The flash creates stronger gradients in well-lit regions, while shadows appear more subdued. This implies raw gradient magnitudes are sensitive to lighting changes.
However, in Figures 3(c) and 3(d), the normalized HOG features demonstrate increased consistency between the flash and no-flash cases. The key edges, particularly around the toys and surface, remain relatively stable despite the lighting change. Normalization ensures that gradient magnitudes do not dominate due to local brightness differences, making the representation more robust.
Conclusion: Normalization is beneficial in reducing the effect of illumination changes, leading to more consistent feature representation. This is particularly useful in object detection tasks where lighting conditions vary. Visually, the normalized HoG retains structural details while reducing the contrast-induced variations seen in the raw HOG. A quantitative measure, such

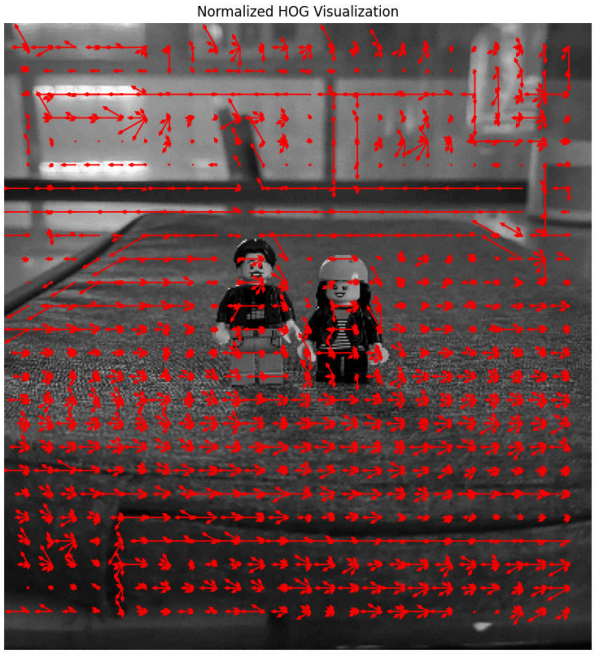
as cosine similarity between the histograms, could further confirm this by showing higher similarity between the normalized HOGs of both images.



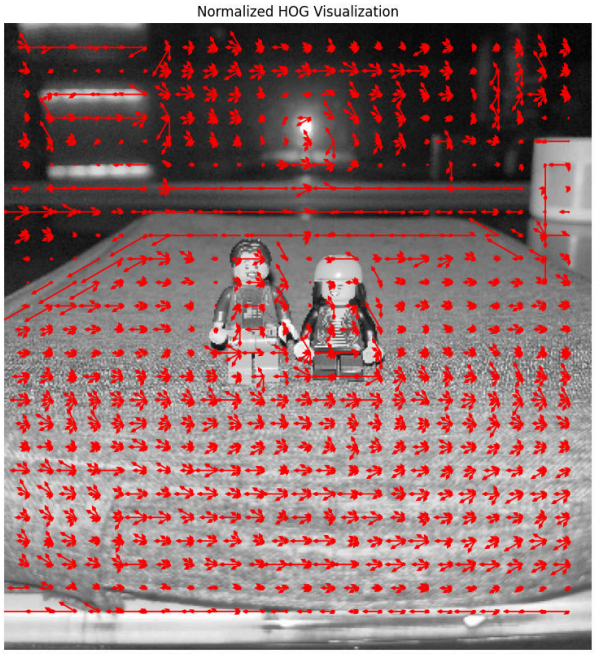
(a) My image without flash using the original (Magnititude) HoG.



(b) My image with flash using the original (Magnititude) HoG.



(c) My image without flash using the normalized (Magnititude) HoG.



(d) My image with flash using the normalized (Magnititude) HoG.

Figure 3: Combined plots of HoG using Magnitudes and Counts.

[Question 4] Corner Detection (40 marks)

Download two images (I_1 and I_2) of the Sandford Fleming Building taken under two different viewing directions:

- https://commons.wikimedia.org/wiki/File:University_College,_University_of_Toronto.jpg
- https://commons.wikimedia.org/wiki/File:University_College_Lawn,_University_of_Toronto,_Canada.jpg

1. Calculate the eigenvalues of the Second Moment Matrix (M) for each pixel of I_1 and I_2.
2. Show the scatter plot of λ_1 and λ_2 for all the pixels in I_1 **(5 marks)** and the same scatter plot for I_2 **(5 marks)**. Each point shown at location (x, y) in the scatter plot, corresponds to a pixel with eigenvalues: $\lambda_1 = x$ and $\lambda_2 = y$.
3. Based on the scatter plots, pick a threshold for $min(\lambda_1, \lambda_2)$ to detect corners. Illustrate detected corners on each image using the chosen threshold **(10 marks)**.
4. Constructing matrix M involves the choice of a window function $w(x, y)$. Often a Gaussian kernel is used. Repeat steps 1, 2, and 3 above, using a significantly different Gaussian kernel (i.e. a different σ) than the one used before. For example, choose a σ that is significantly (e.g. 5 times, or 10 times) larger than the previous one **(10 marks)**. Explain how this choice influenced the corner detection in each of the images **(10 marks)**.

Note that all plots are available in "./assignment3/task2/assets/". Scatter plots (part 2.) are shown in Figure 7. It is clear that for the first image, we should pick a threshold at most 5,000, and more reasonably, 1,500. Which is picked. And for the second image, we can pick 10,000.

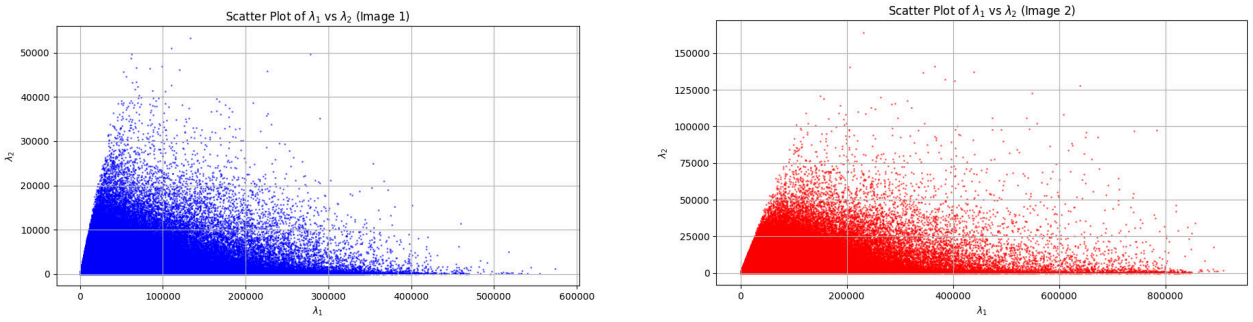


Figure 4: Scatter plots for the pictures of SF for $\sigma = 0.5$.

The corners are illustrated below.

Figure 5: Corner plots for the pictures of SF for $\sigma = 0.5$.

I initially ran for $\sigma = 0.5$, the I did it for $\sigma = 5$, and it gave the below plots. My code generates the corner plots automatically and since I didn't yet change the threshold, it picked up on way more corners. This indicates that we need to increase the threshold to suppress more features. I did a 10x increase in σ, and tried doubling the thresholds to 3,000 and 10,000. This produced similar intensity of corners detected. This is shown in the plot below.

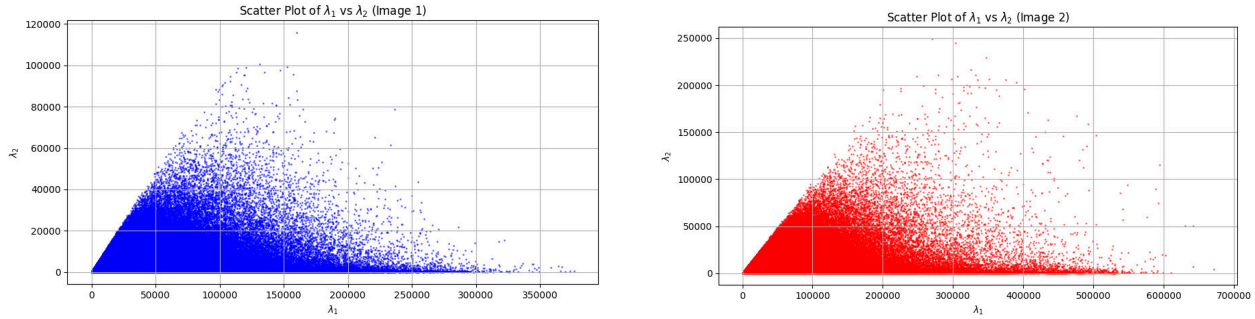


Figure 6: Scatter plots for the pictures of SF for $\sigma = 5$.

Figure 7: Corner plots for the pictures of SF for $\sigma = 5$.

This experiment showed that increasing sigma by a large scale, 10x here, leads to a lesser

increase in the threshold required for corner detection. This aligns well with intuition as the window function is taken for a range, and as σ gets so big that it is the entire window, then increasing it further does not lead to much difference. In fact, I ran it with $\sigma = 500$ to see, and it produced the exact same figures as $\sigma = 5$. I think the idea behind this is that the gaussian kernel with $\sigma = 500$, is effectively the same as the gaussian kernel with $\sigma = 5$ in the range the window function is defined. Moreover, the smaller $\sigma = 0.5$, spans less and thus changing to $\sigma =$ leads to a noticable effect.