Computer Vision *ECS 174, Fall 2025*

HW1: Filters and edge detection

Please turn in the complete code and its results. It may be a notebook or zipped code with a PDF.

Keep in mind that any extra points earned can only boost your grade for this specific assignment, up to the maximum points possible. However, these additional points cannot make up for any missed points in other assignments.

Important:

1. This is a one-person exercise. You are more than welcome to reach out to TA or your friends and teammates to seek help with questions, but plagiarism in any form will not be tolerated.
2. Using ChatGPT or any other large language model in answering these questions is against the Code of Conduct and will be reported.
3. Your codes should run in under 5 minutes before producing the results.

Before you start, please download the project's files from the course webpage.

For this assignment, you'll dive into implementing functions for filtering and edge detection. It's important to note that Python offers built-in functions for many of the tasks you'll undertake here. However, for educational purposes, refrain from utilizing those built-ins in your final code. Avoid using filtering commands and similar functionalities directly. Instead, focus on implementing the logic from scratch. Feel free to use built-in functions only for result verification purposes. Also, you are not allowed to use ChatGPT or similar methods in solving any assignment of this course.

1. (25 points) Implement the function GaussianBlurImage(image, sigma) to Gaussian blur an image. "sigma" is the standard deviation of the Gaussian. Implement the Gaussian blur using a 2D filter.

   Required: Gaussian blur the image "Seattle.jpg" with a sigma of 4.0, and save as "1.png".

2. (25 points) Implement the function SeparableGaussianBlurImage (image, sigma) to Gaussian blur an image using separable filters. "sigma" is the standard deviation of the Gaussian. The separable filter should first Gaussian blur the image horizontally, followed by blurring the image vertically. The final image should look the same as when blurring the image with GaussianBlurImage.

   Required: Gaussian blur the image "Seattle.jpg" with a sigma of 4.0, and save as "2.png".

3. (25 points) Implement SobelImage(image) to compute horizontal and vertical edges, edge magnitude, and edge orientation information. SobelImage should display the outputs in an image. You may convert color image into a gray scale image and then run the filter on that.

   Required: Compute Sobel edge filter on "LadyBug.jpg" and save as 5a, 5b, 5c, and 5d ".png". You may use `cmap = plt.cm.hsv` to save orientation and `cmap='gray'` to rest of images.

4. (25 points) Implement BilinearInterpolation(image, x, y) to compute the linearly interpolated pixel value at (x, y). Both x and y are continuous values.

   Required: Upsample image "Moire_small.jpg" to be 4 times larger in each direction (16 times more image area) once with nearest neighbor interpolation and save as "6a.png" and once with bilinear interpolation and save as "6b.png".

5. **(25 extra points)** Implement BilateralImage(image, sigmaS, sigmaI) to bilaterally blur an image. "sigmaS" is the spatial standard deviation and "sigmaI" is the standard deviation in intensity. Hint: The code should be very similar to GaussianBlurImage.