CS 558: Homework Assignment 1
Due: October 3, 6:00pm

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
pmordoha@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment. I will assume that, as participants in a graduate course, you will be taking the responsibility of making sure that you personally understand the solution to any work arising from collaboration.

Late Policy. A total of 3 late-days will be allowed in the entire semester without penalty. Days will be counted by rounding up, e.g. being 5 minutes late will be counted as one of the late-days. Assignments will no longer be accepted after all late-days have been used.

Deliverable. Download the skeleton code provided with the assignment, complete and **execute all cells.** Then, save the **Jupyter notebook** and submit it on Canvas. Please read the following list carefully:

- You should not change the provided function specifications.
- You are not allowed to use **any** library functions that operate on the images other than to read and write the files and to display images on the screen. (See the NumPy tutorial for examples.) Math functions from NumPy are allowed, but no filtering, convolution, resizing, edge detection etc. functions that have not been implemented by you may be used.

Guidelines and Hints

- Download the images provided in the assignment and try your code on them, but keep in mind that we may test it on additional images. Do **not** submit a spaghetti version of the code operating on multiple images. The input file name should be specified once in `main()` and all operations should be carried out on the same image.
- Images should be converted to gray scale.
- Do not worry about the path to the input data.

- If you develop using a single Python script, you still have to submit a notebook by populating the boxes. Under the *Runtime* menu, there is an option for executing all cells.
- Do not activate *playground mode* so that your changes to the notebook are preserved.
- Since you are given the entire structure of the code this time, attention to detail is what is being evaluated.

Step 1: the Sobel filter. (20 points) Apply the horizontal and vertical Sobel filters on a grayscale image. The dimensions of the output must be the same as those of the input image. Use **zero-padding** to handle boundaries. **Display** images with the absolute values of the two responses (horizontal and vertical) separately. (No need to save the files.)

Step 2: the Gaussian filter. (20 points) Now apply a Gaussian filter to the input image, using the same fltering function you implemented above with zero-padding. Allow the user to specify the σ of the Gaussian function in `main()`. Make sure that the filter size is large enough and that the filter coefficients sum to 1. The `main()` function should **display** the kernel as a small image and the output filtered image. Try values of σ ranging from 1 to 10.

Step 3: both filters. (20 points) Using the above functions, low-pass filter the image and then **display** the magnitude of the image gradient.

If you are having trouble with the Gaussian filter, use a mean filter (box of constant coefficients) to complete this and the next steps if Step 2 fails.

Step 4: the Harris operator. (20 points) Compute and display the magnitude of the Harris operator. The window size should be a parameter. You can use 5×5 as the default window size - this is not the Gaussian filter from Step 3. The weights should be constant. This can be implemented using the filtering function or by averaging within this function.

Step 5: non-max suppression. (20 points) Apply non-max suppression to the output of the Harris operator in 3×3 neighborhoods. Keep in mind that the corners we are looking for are isolated points. Threshold the local maxima to keep only the most important corners, aiming for 100-300 corners per image. (The threshold will depend on the input image, so there is no need to worry too much about this step.) **Display** a binary (black and white) image with the corners only and also super-impose the corners on the input using a technique of your choice.