First Submission due on Sunday, February 9th at 11:59 pm
Answer all problems following the instructions. You must use the Latex template for your solutions. Please compile the Latex template and submit the PDF report to Canvas.

Instruction Guidelines:

Please use the Latex template for your solutions. Presentation is very important (and worth 10% of the grade). For every problem, you will write up your solution, assign yourself a numerical grade **with reasons/justification for the grade**, and acknowledge any web sources, classmate help that you used in this assignment. Note: ChatGPT usage is allowed for problems, but you are responsible for ensuring the validity of its solutions and justifications. Also note that writing of the report can also use genAI tools, but your reasons/justifications/explanations in the report should be backed up with evidence (figures, numerical calcuations, etc.) wherever possible.

Problem 0: *Setting up your Coding Environment:*
We will utilize the Python programming language along with OpenCV bindings to do most of the image processing required in the homework assignments. In addition, we will utilize several helper libraries for visualizing images and displaying output. Problem 0 is not graded but rather provided to make sure you install OpenCV successfully.

1. Install Anaconda Python `https://docs.anaconda.com/anaconda/install/`. Make sure you follow the instructions specific to your operating system. I used the graphical interface to install, although you can also use the command line version if you are comfortable with that. To test your Python is working, create a file **helloworld.py**, and write the following line:

```
print('Hello World!')
```

   To run the code, you can open a terminal shell and run the following command:

```
python helloworld.py
```

   and should get the terminal output of "Hello, World!". If you run into errors, check your installation of Anaconda Python (Google is your friend for debugging errors! and StackOverflow).

2. Install OpenCV via Anaconda using the following command:

```
conda install -c conda-forge opencv
```

   If this command does not work, Google how to install OpenCV for your Anaconda distribution (or search for OpenCV on the package finder)

3. Install Matplotlib (a useful library for visualizing graphical output)

```
conda install matplotlib
```

4. Make sure you can import all the libraries. Write the following file **testimport.py** with the following lines:

```
import cv2
import matplotlib
import ipdb
```

   If you get no errors while running this, then you have successfully installed these packages. **NOTE:** If you are stuck on any step, Google is your friend! Just type "How do I install X on a Mac/Windows/Linux?" and experiment till you find something that works!

5. I recommend using an IDE such as Spyder `https://www.spyder-ide.org/` or PyCharm `https://www.jetbrains.com/pycharm/` to prototype your code. Spyder is built into Anaconda Python, just use the package installer in Anaconda to install it.

Problem 1. *Messing around with OpenCV and Images* (10 points)
The point of this problem is to mess around with images, and explore the OpenCV framework.

(a) Pick an image you really like to display, and use the package Matplotlib to do so. Commands such as *plt.imshow* and *plt.show* are helpful here. Notice how the colors seem inverted, so you can use the function *cv2.cvtColor* to change that.

(b) Zoom in on a particular part of the image by cropping the image. Display the image. In the report, explain what you are looking at in this part of the image. Comment about the textures, the colors being displayed, etc. The point of this question is to get you looking deeply at images and making observations.

(c) Read in another image to be used for image sampling. Downsample the image by 10× in width and height. Use the *cv2.resize* command. If you have problems with this command, look at examples online and use the keyword "None" for the optional parameters. Display the downsampled image.

(d) Using the resize command, upsample the same downsampled image from part b by 10x back to its original resolution. This time, try two different interpolation methods: nearest neighbor and bicubic. Display them in the report.

(e) Calculate the absolute difference between the ground truth image and the two upsampled images with the two methods (find the appropriate OpenCV command, should be one line of code for each image). Write out the difference images for both methods. Then sum all the pixels in the difference image for the two methods, and report the number. Which method caused less error in upsampling? Explain when you might use one method over the other (you should have use cases for using both methods, not just using one method all the time).

Problem 2. *(Convolution as Matrix Multiplication and Edge Filtering)* (20 points)

We will implement 2D convolution as a matrix multiplication. In fact, this is used often in machine learning algorithms such as convolutional neural networks where one needs to perform

several convolutions for each layer/filter bank of the algorithm. We will now implement 2D convolution as a matrix operation. Define h as follows:

$$h = \begin{bmatrix} 1 & 0 & -1 \\ 1 & \mathbf{0} & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

.

(a) First use image

$$I = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

. Assume zero-padding (i.e. full padding with zeros), and you can choose whether or not you wish to flip the kernel. First, vectorize $\vec{I}$ where each row of the image is stacked vertically into a column vector, i.e. $\vec{I} = [1, 2, 3, 4, 5, 6, 7, 8, 9]^T$. Then write by hand a matrix version of the kernel h, denoted as capital H such that $H * \vec{I}$ is the vectorized output image (i.e. $H * \vec{I} = h * I$). H should be a 25 x 9 matrix in this case. The output should be a $25x1$ vector, that corresponds to a 5×5 output image that is row stacked. This part should be calculated by hand on paper, but you can use a computer to check your answer.

(b) Now write a function *conv2dmatrix* that takes inputs $[image, H]$ and outputs $[image * H, time]$. Time here refers to the latency of the method. This code should compute the same result as (a) which you did calculate numerically.

(c) Generalize your code so both the image and the convolution kernel is any of the user's choosing, and the new function can still compute the convolution as a matrix multiplication. The output should be an edge filtered image. Specify how you handled the dimension issues in your implementation.

(d) Now using your convolution function, first implement a baseline version of the Canny Edge Filter (See the Wiki page: `https://en.wikipedia.org/wiki/Canny_edge_detector` for a description of the main algorithm to implement). Implement a version of your own (Note: you must implement from scratch, do not use a built-in package), and then **compare your version versus OpenCV's built-in Canny edge detector.** Analyze the performance differences between the two implementations and show some comparative figures to illustrate your points.

(e) Finally, **propose your own novel variant of a Canny edge detector**. Push yourself to try something new, and document what you find. Compare your new detector (give it a name!) against the built-in OpenCV Canny edge detector. Note: you will be graded in this problem on creativity, and showing your development process. If you give us something ChatGPT could give us, that makes you irrelevant as the computer vision developer and we will award ChatGPT the points :)

Problem 3. *(Fourier Domain Fun)* (10 points)

(a) Read in a picture of your choice, and make it grayscale. Plot the Fourier transform (magnitude and phase) of the picture. Discuss why the Fourier transform looks the way it does, comment on certain key features in the Fourier space and why they appear that way (or what might be the cause of them in the spatial domain).

(b) Implement both a low-pass, high-pass filter, and a diagonal bandpass filter (of your choice) on the picture in the frequency domain. Make sure to plot the Fourier magnitudes of the image before and after filtering. Show the resulting filtered images.

(c) Phase Swapping: Select two images, compute their Fourier transform, and display their magnitude and phase images. Then swap their phase images, perform the inverse Fourier transform, and reconstruct the images (display these). Comment on how this looks perceptually. Try to modify the phase differently than just swapping them, and explain what observation you made.

(d) Hybrid Images: Take two images (of your choice), and try the hybrid images approach from the paper from Olivia et al `http://olivalab.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf`. Explain a set of rules/guidelines for what images are best to merge with the hybrid images approach, and show evidence for your reasoning with actual image examples. I will show the best hybrid image results in class!

Problem 4. *(Multiresolution Blending using Gaussian/Laplacian Pyramids)* (20 points)

In this problem, you are going to practice performing multiresolution blending. Use the paper by Burt and Adelson (`http://ai.stanford.edu/~kosecka/burt-adelson-spline83.pdf`) as reference for this assignment.

(a) First pick two or more images that you wish to use for blending. Please be creative here in your choice of images and why you want to blend them. In the report, please explain your choice of images and the type of blending you want to do, and why you felt multiresolution blending would work for this problem. Note: to prototype your algorithm is working properly, you can utilize the original orange/apple example from the paper to check your results as well.

(b) First write a code to generate the Gaussian and Laplacian pyramids of an image. You should confirm that your Laplacian pyramid can be collapsed back into the original image (both qualitatively and quantitatively). You can use existing OpenCV commands as you want, but please show that your method is working properly.

(c) Make mask(s) that will act as the transition between your images (binary masks).

(d) Direct Blending: Try to directly blend the images by using $I = (1 - M) * I_1 + M * I_2$, where M is the mask and $I_{1,2}$ are the two images. If you have multiple images, you can use this formula multiple times as you stitch images together. Display this image.

(e) Alpha Blending: Try to blur the mask edge with a Gaussian $G(M)$, and then $I = (1-G(M))*I_1 + G(M) * I_2$ (i.e. alpha blending or feathering). How does this look (does it look like one fruit?)

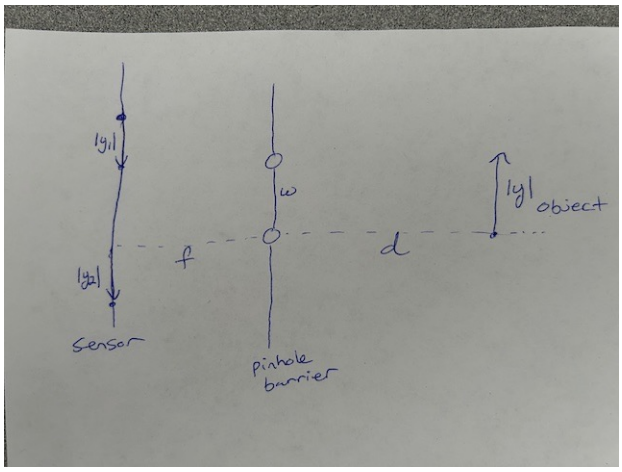

Figure 1: Figure for Problem 5

(f) Multiresolution blending: Write a function *multiblend* that takes in as inputs [*image*1, *image*2] and outputs [*blendedimage*]. Construct a Gaussian pyramid of the Mask, and Laplacian pyramids of the images. Blend the images using the multiresolution blending algorithm shown in class. Try to play around with parameters (such as depth of pyramids), etc until you find a blending you like. In your report, discuss which of the three blendings (direct, alpha, multiresolution) worked the best for your imaging examples, and speculate why that might be the case. Please show at least three different sets of images (that are diverse) for blending to support your claims.

Problem 5. *Theory problem: Dual Pinholes* (10 points)

(a) Given Figure 1, derive the optimal pinhole width $w*$ given a fixed f and d and object $|y|$ such that there is exactly two copies of the image right on top of one another with no blurring (i.e. no distance between the two images, but no overlap either). Show your work for your calculation.

Problem 6. *Pinspeck Cameras* (20 points)
We are going to explore the anti-pinhole camera or pinspeck camera, and show some applications.

(a) Read the paper at the project website: "Accidental Pinhole and Pinspeck Cameras" by Antonio Torralba and William T. Freeman `https://people.csail.mit.edu/torralba/research/accidentalcameras/`.

(b) Write code to load in the movie "livingroom.mov" provided in webpage. The OpenCV function *cv2.VideoCapture* will help read in movies and convert them to frames.

(c) Print out the difference video, achieving something similar to the video "livingroomoutput.avi". Talk about the choice of reference frame you use for subtraction (Torralba/Freeman

discuss this in the paper, mention which strategy you use.) Please choose appropriate figures to visualize your results.

(d) Isolate a portion of the video that corresponds to the pinspeck image. See if you can match the scene outside the window that looks like "outdoorimage.png".

(e) Using your own camera/phone, try to recreate a pinspeck image with a window or aperture. Show your original background image and image with occluder, the difference image, and the ground truth image outside the window. Note I'm not looking for perfection here, just a good effort at capturing a pinspeck image. You can see if you can acquire RAW images from your cell phone, or borrow a DSLR camera from a friend if you want to improve the quality.

1 Grading and Report

Grading breakdown is as follows:

- Problem 1: 10 points
- Problem 2: 20 points
- Problem 3: 10 points
- Problem 4: 20 points
- Problem 5: 10 points
- Problem 6: 20 points
- Presentation (are answers clearly defined and easy to find, is code snippets and figures utilized well for the report): 10 points