Problem Set 1 – CS 4476: Fall 2025

Instructor: Judy Hoffman

Due: September 2, 2025

Instructions

1. **Collaboration Policy:** You are encouraged to discuss this assignment with others, BUT your code and writeup must be completed individually. If we find highly identical write-ups or code we will take action according to strict university policies. See the Academic Integrity Section on the course syllabus for more information.

2. **Late Submission Policy:** There are a **total of 5** late days across all homework submissions. Submissions that use additional late days will incur a 10% penalty per late day. No assignments will be accepted more than a week after the due date.

3. **Submitting your work:**

   - **Report:**
     - Fill your answers in the report PPT file provided (`FirstName_LastName.pptx`). Submit a PDF version of it with your name to Gradescope for the assignment **PS1: Report**. Ensure your answers fit within the space provided.
     - Do not modify the layout of the boxes provided in the report.
     - If plots are required, you must include them in your Gradescope report and your code must display them when run. Points will be deducted for not following this protocol.
   - **Code:**
     - Enter your code in the designated areas of the template Python files.
     - The assignment must be done in Python3. No other programming languages are allowed.
     - Submit your code and output files in a zipped format to Gradescope for the assignment **PS1: Code**. To properly zip use the helper script with your GT username as below.

       ```
       > python zip_submission.py --gt_username <your_gt_username>
       ```

       The `.zip_dir_list.yml` file contains the required deliverable files, and `zip_submission.py` will fail if all the deliverables are not present in the root directory. Feel free to comment and uncomment them as you complete your solutions. **Do not create subdirectories within the main directory.**
     - Make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.

4. **Formatting:** Our assignments are specifically templated to facilitate prompt grading of your work. To enable this process, it is essential that you do not modify the formatting of the report or code templates. **Modifications will incur a penalty.** This means:

   - Do not move the boxes in the pptx report template.
   - Do not add new subfolders to the code.
   - Do not use absolute file paths in your code.

- Do not add additional functions/imports to the files.
- Ensure that you follow the instructions very carefully. Refer to the Deliverable Checklist at the end of these instructions to make sure you have completed everything.

This assignment is 100 points total: 50 Report + 50 Code.

Setup

Note that we will be using a new conda environment for this project.

1. Install Miniforge. Refer to the Mac/Linux instructions or the Windows instructions. This will install the conda and mamba commands on your machine.
2. Open the terminal
   (a) On Windows: open the installed **Miniforge Prompt** to run the command
   (b) On MacOS: open a terminal window to run the command
   (c) On Linux: open a terminal window to run the command
3. Navigate to the project directory
4. Create the conda environment for this project
   (a) On Windows: `mamba env create -f proj1_env_win.yml`
   (b) On MacOS: `mamba env create -f proj1_env_mac.yml`
   (c) On Linux: `mamba env create -f proj1_env_linux.yml`
5. Activate the newly created environment, using the command `conda activate cs6476_proj1`

Introduction

Read through the provided Python, NumPy and Matplotlib introduction code and comments here or here (thank you, web archive). You may also find official documents of numpy and matplotlib helpful.

Open an interactive session in Python and test the commands by typing them at the prompt. (Skip this step if you are already familiar with Python and NumPy.)

After reading the required documentation in the above section, to test your knowledge, ensure that you know the outputs of the following commands without actually running them.

```
> import numpy as np
```

(a)
```
> x = np.random.permutation(1000)
```

(b)
```
> a = np.array([[11,22,33],[40,50,60],[77,88,99]])
> b = a[2,:]
```

(c)
```
> a = np.array([[11,22,33],[40,50,60],[77,88,99])
> b = a.reshape(-1)
```

(d)
```
> f = np.random.randn(5,1)
> g = f[f>0]
```

(e)
```
> x = np.zeros(10)+0.5
> y = 0.5*np.ones(len(x))
> z = x+ y
```

(f)
```
> a = np.arange(1,100)
> b = a[::-1]
```

1 Basic Numpy [12 points (3 each)]

Write a few lines of code to do each of the following. See the report powerpoint for starter code with function definitions. Copy and paste your code into the designated place in the report.

1.1 Use `numpy.random.rand` to return the roll of a six-sided die over N trials.

1.2 Let `y` be the vector: `y = np.array([11, 22, 33, 44, 55, 66])`. Use the reshape command to form a new matrix `z` that looks like this: `[[11,22],[33,44],[55,66]]`

1.3 Use the `numpy.max` function to set `x` to the maximum value that occurs in `z` (above), and use the `numpy.where` function to set `r` to the row number (0-indexed) it occurs in and `c` to the column number (0-indexed) it occurs in. You may assume the maximum value is unique.

1.4 Let `v` be the vector: `v = np.array([1, 4, 7, 1, 2, 6, 8, 1, 9])`. Set a new variable `x` to be the number of 1's in the vector `v`.

2 Array/Matrix Operations [20 points (4 each)]

Load the 100x100 matrix `inputAPS1Q2.npy` which is the matrix A. Fill the template functions in the script `PS1Q2.py` to load `inputAPS1Q2.npy` and perform each of the following actions on `A`.

2.1 Plot all the intensities in `A`, sorted in decreasing value. Provide the plot in your report. (Note, in this case we don't care about the 2D structure of `A`, we only want a sorted list of all the pixel intensities in `A`.) Use the "gray" colormap in matplotlib. (Complete function `prob_2_1` and provide the plot in your report.)
Hint : Set the *aspect* argument in matplotlib's `imshow` function appropriately

2.2 Display a histogram of `A`'s intensities with 20 bins. Again, we do not care about the 2D structure. (Complete function `prob_2_2` and provide the histogram in your report.)

2.3 Create and return a new matrix `X` that consists of the bottom left quadrant of `A`. (Complete function `prob_2_3`.)

2.4 Create and return a new matrix `Y`, which is the same as `A`, but with `A`'s mean intensity value subtracted from each pixel. (Complete function `prob_2_4`.)

2.5 Create and return a new matrix `Z` that represents a color image with the same size as `A`, but with 3 channels to represent R, G and B values. Let a threshold `t` = the average intensity in `A`. In the new matrix `Z`, set the pixel value to be red (i.e., R = 1, G = 0, B = 0) in locations where the intensity in `A` is greater than `t`, and set the pixel value to be black everywhere else. (Complete function `prob_2_5`.)

3 Image Manipulations [36 points (6 each : 4 Code + 2 Report)]

The input color image `inputPS1Q3.jpg` has been provided. Fill the template functions in the script `PS1Q3.py` to perform the following transformations on this image. Avoid using loops. Provide all the resultant images in your report. Use the "gray" colormap when displaying images with a single color channel (e.g. grayscale images).

Note: In every part, the image that is returned from your function must have integer values in the range [0, 255] (i.e `uint8` format). *Tip:* perform calculations with the image in double type and cast it to integer type before displaying and returning it.

Note: If you use OpenCV (cv2), be aware that this library uses the channel order convention BGR rather than RGB. skimage uses RGB ordering.

3.1 Load the input color image and swap its red and green color channels. Return the output image. (Complete function `prob_3_1` and provide the swap image in your report.)

3.2 Complete the helper function `rgb2gray` to convert a color image to a grayscale image. You can use the formula Gray = 0.2989 * Red + 0.5870 * Green + 0.1140 * Blue. In `prob_3_2`, convert the input color image to a grayscale image by calling your `rgb2gray` function. Return the grayscale image. (Complete function `prob_3_2` and provide the grayscale image in your report.)

For parts 3.3 - 3.6, perform each of the following transformations on the grayscale version of input image you produced in part 3.2. You may do this by calling your `rgb2gray` function in each part. When plotting the resulting images, make sure you use the "gray" colormap option. Additionally, several problems in the rest of this assignment may require handling of overflow issues when adding/multiplying (ex: Problem 3.5). Make sure to typecast the image from `uint8` to `float` or `double` when appropriate.

3.3 Convert the grayscale image to its negative image, in which the lightest values appear dark and vice versa. Return the negative image. (Complete function `prob_3_3` and provide the negative image in your report.)

3.4 Map the grayscale image to its mirror image (flipping it left to right). Return the mirror image. (Complete function `prob_3_4` and provide the mirror image in your report.)

3.5 Average the grayscale image with its mirror image. Return the averaged image. (Complete function `prob_3_5` and provide the average image in your report.)

3.6 Create a matrix `noise` whose size is same as the grayscale image, containing random numbers in the range [0, 255]. Add `noise` to the grayscale image, then clip the resulting image to have a maximum value of 255. Return the clipped image and the noise matrix. (Complete function `prob_3_6` and provide the clipped noisy image in your report.)

4 Understanding Color [19 points]

4.1 The same color may look different under different lighting conditions. Images `indoor.png` and `outdoor.png` are two photos of a same Rubik's cube under different illuminances. Load the images and plot each of their R, G, and B channels separately as a grayscale image for each channel. Then, convert them into LAB color space using `cv2.cvtColor()` or `skimage.color` and plot the three channels (L, A, and B). There will be 12 total subplots: for each of the 2 images (indoor and outdoor), you are plotting the 3 RGB channels and the 3 LAB channels. Be sure to label your plots appropriately, including specifying which subplot corresponds to which channel. Use the "gray" colormap for plotting each channel. Include the plots in your report. (Complete function `prob_4_1` and include the plots in the report.) [points: 7 Report]

4.2 Read the section below to understand the process of translating the cubical colorspace of RGB to the cylinder of hue, saturation, and value (HSV). Read through the explanation and fill in the template function in `PS1Q4.py` to load `inputPS1Q4.jpg`, convert the image from RGB to HSV, and return the final HSV image. Plot the HSV image using matplotlib's `imshow` function and include it in the report. You may use for loops for this question. You are not allowed to use a library function to do the conversion. (Complete function `prob_4_2` and provide the HSV image in the report.) [points: 7 Code + 5 Report]

Note: To ensure consistency, do the necessary typecasting (`double`) and transform the image to values between `[0,1]` before performing the operations to convert to HSV.

So far we've been focusing on RGB and grayscale images. But there are other colorspaces out there too we may want to play around with. Like Hue, Saturation, and Value (HSV).

Hue can be thought of as the base color of a pixel. Saturation is the intensity of the color compared to white (the least saturated color). The Value is the perception of brightness of a pixel compared to black. You can try out this demo to get a better feel for the differences between these two colorspaces.

Now, to be sure, there are lots of issues with this colorspace. But it's still fun to play around with and relatively easy to implement. The easiest component to calculate is the Value, it's just the largest of the 3 RGB components:

$$V = max(R, G, B)$$

Next we can calculate Saturation. This is a measure of how much color is in the pixel compared to neutral white/gray. Neutral colors have the same amount of each three color components, so to calculate saturation we see how far the color is from being even across each component. First we find the minimum value

$$m = min(R, G, B)$$

Then we see how far apart the min and max are:

$$C = V - m$$

and the Saturation will be the ratio between the difference and how large the max is:

$$S = C/V$$

Except if R, G, and B are all 0. Because then V would be 0 and we don't want to divide by that, so just set the saturation 0 if that's the case.

Finally, to calculate Hue we want to calculate how far around the color hexagon our target color is.

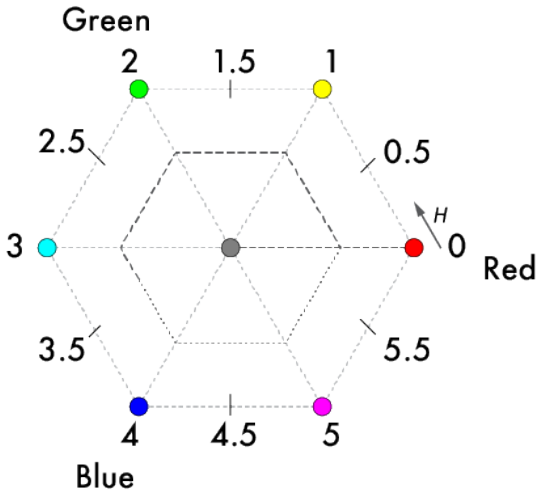

Figure 1: Color Hexagon

We start counting at Red. Each step to a point on the hexagon counts as 1 unit distance. The distance between points is given by the relative ratios of the secondary colors. We can use the following formula from Wikipedia:

$$H' = \begin{cases} undefined & \text{C} = 0 \\ \frac{G-B}{C} & \text{if V} = \text{R} \\ \frac{B-R}{C} + 2 & \text{if V} = \text{G} \\ \frac{R-G}{C} + 4 & \text{if V} = \text{B} \end{cases}$$

$$H = \begin{cases} \frac{H'}{6} + 1 & \text{if H'} < 0 \\ \frac{H'}{6} & \text{otherwise} \end{cases}$$

There is no "correct" Hue if C = 0 because all of the channels are equal so the color is a shade of gray, right in the center of the cylinder. However, for now let's just set H = 0 if C = 0 because then your implementation will match ours.

5 Hybrid Images [13 points]

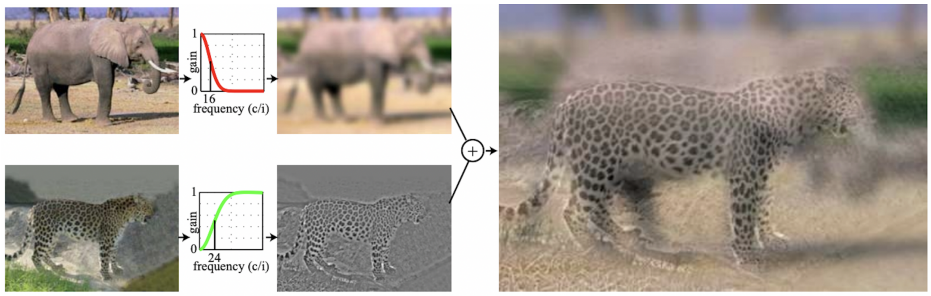

Figure 2: Hybrid images are created by filtering one image with a low-pass filter and another image with a high-pass filter. The hybrid image is generated by adding the two filtered images together.

In this question, you will apply filters to images and use a low pass filter to create hybrid images (Oliva et al., 2006). Hybrid images are created by combining the low frequency content of one image with the high frequency content of another image. Take a look at Figure 2, which combines the low frequency content of the top image (an elephant) with the high frequency content of the bottom image (a leopard). The result is a new image that looks more like the leopard when viewed up close, and more like the elephant when viewed from far away!

You will create a hybrid image from 2 provided color images, `cat.bmp` and `dog.bmp`. Fill in the template functions in the script `PS1Q5.py` and submit this file. Provide all resultant images in your report.

5.1 Load the input color images. In this problem, you will need to convert the image values to floating point values between 0 and 1. Then, complete the function `my_conv2d` to apply a 2d convolution to an image using a given filter. You will use `scipy.ndimage.convolve` with the default parameters on each channel of the image. Then, in `prob_5_1`, use your `my_conv2d` to apply each of the following filters to `cat.bmp`: Identity, Blur, Sobel, and Laplacian. These filters are given to you as `self.identity_filter`, `self.blur_filter`, `self.sobel_filter`, and `self.laplacian_filter`. Note that the Sobel filter is implemented only for the x-direction and we do not expect your solution to contain the y-direction. Include the 4 filtered images in your report. [points: 4 code + 4 report]

5.2 Create a hybrid image from `dog.bmp` and `cat.bmp` by combining the low frequency of the dog image with the high frequency content of the cat image. Do this in `create_hybrid_image`. You will use your `my_conv2d` function with a Gaussian low pass filter given by `self.gaussian_filter`. Then, in `prob_5_2`, use matplotlib's `imshow()` to display your hybrid image. Additionally, use the given `vis_image_scales_numpy()` function to generate a visualization of your hybrid image at different scales. Include both your hybrid image and this visualization in the report. [points: 3 code + 2 report]

Deliverable Checklist

1. 1.1-1.4 (report) code snippets for each question.

2. 2.1-2.5 (code/files) `PS1Q2.py`, (report) corresponding plots for 2.1 & 2.2.
3. 3.1-3.6 (code/files) `PS1Q3.py`, (report) 6 images.
4. 4.1 (report) display the channel-wise RGB and LAB plots for the 2 images
5. 4.2 (code/files) `PS1Q4.py`, (report) HSV image.
6. 5.1 (code/files) `PS1Q5.py`, (report) 4 filtered images
7. 5.2 (code/files) `PS1Q5.py`, (report) hybrid image and multi-scale visualization

Submission Instructions

- Submit the code as zip on Gradescope at **PS1: Code**
- Submit the report as PDF on Gradescope at **PS1: Report**

There is no submission to be done on Canvas.

This assignment is adapted from the following 4 sources:

PS0 assignment of Kristen Grauman's CS 376: Computer Vision at UT Austin

HW1 assignment of David Fouhey's EECS 442: Computer Vision at University of Michigan. (This should be a similar one.)

HW0 assignment of Joseph Redmon's CSE 455: Computer Vision at University of Washington.

Project 1 of James Hays's CS 4476/6476: Computer Vision at Georgia Tech.