Problem Set 1: Images as Functions

January 9, 2025

ASSIGNMENT DESCRIPTION

Description

Problem Set 1 is a warm-up assignment to get you started with loading images in Python, manipulate the values, produce some outputs, and submit the code along with the report to Gradescope. Note that autograded problems will be marked with a (*). It is expected that you have set up your environment properly. All problem sets will require the following libraries: NumPy, SciPy and OpenCV. If you are having problems, look for information on Ed discussion, as someone may have resolved it, if not, post a question with detailed specifics of your problem. We want to emphasize that the goal of this problem set is to get you all set up so we can do more in later weeks.
Please do not use absolute paths in your submission code. All paths should be relative to the submission directory. Any submissions with absolute paths may receive a penalty!

Learning Objectives

- Learn to load, display, and save images.
- Study how images can be represented as functions.
- Identify the difference between an RGB and Monochrome / Grayscale images.
- Apply linear algebra concepts to manipulate image pixel values.
- Perform basic statistical operations in arrays.
- Introduce the concept of noise in an image.

Starter Code

Obtain the starter code from canvas under files.

INSTRUCTIONS

Programming Instructions

Your main programming task is to complete the functions described in the file **ps1.py**. The driver program **experiment.py** helps you illustrate the results and will output the files needed for the writeup.

Write-up Instructions

Create **ps1_report.pdf** - a PDF file that shows all your output for the problem set, including images labeled appropriately (by filename, e.g. ps1-1-a-1.png) so it is clear which section they are for and the small number of written responses necessary to answer some of the questions (as indicated). You are required to use do the following when creating your report:

- Use the LATEX template provided.
- PLEASE be sure to add your name and email on Gradescope, exactly as it is on CANVAS so we can link the two accounts.

How to Submit

Two assignments have been created on Gradescope: one for the report - **PS1_report**, and the other for the code - **PS1_code**.

- Report: the report (PDF only) must be submitted to the PS1_report assignment. assignment-description .unnumbered
- Code: all files must be submitted to the PS1_code assignment. DO NOT upload zipped folders or any sub-folders, please upload each file individually. Drag and drop all files into Gradescope.

Notes

- You can only submit to the autograder **10** times in an hour. You'll receive a message like "You have exceeded the number of submissions in the last hour. Please wait for 36.0 mins before you submit again." when you exceed those 10 submissions. You'll also receive a message "You can submit 8 times in the next 53.0 mins" with each submission so that you may keep track of your submissions.
- If you wish to modify the autograder functions, create a copy of those functions and DO NOT mess with the original function call.

YOU MUST SUBMIT your report and code separately, i.e., one submission for the code and one for the report. Upload each submission to the appropriate Gradescope assignments. By default, your last submission before the deadline will be accounted for and graded. That said, Gradescope also allows you to activate a previous submission if needed. This activated submission will be the one graded.

Grading

The assignment will be graded out of 100 points. The code portion (autograder) represents 60% of the grade and the report the remaining 40%.

ASSIGNMENT QUESTIONS

1.a. Input Images

a. Pick two interesting images to use. Name them ps1-1-a-1.png and ps1-1-a-2.png. Place them in the same directory as the ps1.py file. They should be color, rectangular in shape (NOT square). The first and second images should be wide and tall respectively. You might find some classic computer vision examples here, or you may use your own. Make sure the image width or height each does not exceed 512 pixels and that it is at least 100 pixels.

Code: In the file experiment.py, complete the image paths.
Report: Place your interesting images (wide and tall images) ps1-1-a-1.png and ps1-1-a-2.png in the writeup.

1.b. *Color Planes

a. Swap the green channel and blue channel of image 1
Code: implement swap_green_blue()

b. Make a monochrome image (img1_green) created by selecting the green channel of image 1. (Your monochrome image must be a 2D array)
Code: implement extract_green()

c. Make a monochrome image (img1_red) created by selecting the red channel of image 1. (Your monochrome image should be a 2D array)
Code: implement extract_red()

Report: No write-up for this section

1.c. *Replacement of pixels

Note: For this, use ps1-2-b-1.png from 2-b as your monochrome image.

a. Insert the center square region of 100x100 pixels of the monochrome version of image 1 into the center of a monochrome version of image 2.
Code: implement copy_paste_middle()
Report: No writeup for this section

b. Insert the center circle region with radius 50 pixels of the monochrome version of image 1 into the center of a monochrome version of image2.
Code: Implement copy_past_middle_circle()
Report: No writeup for this section

1.d. *Arithmetic and Geometric operations

a. Compute the min, max, mean, and standard deviation of pixel values in the monochrome image.
Code: implement image_stats()
Report: No writeup for this section

b. Subtract the mean from all pixels in the monochrome image, then divide by standard deviation, then multiply by the scaling factor 10 if your image is 0 to 255 or 0.05 if your image ranges from 0.0 to 1.0. Now, add the mean back into the product.
Code: implement center_and_normalize()
Report: No writeup for this section

c. Shift img1_green to the left by 2 pixels.
Code: implement shift_image_left()
Report: No writeup for this question

d. Subtract the shifted version of img1_green from the original img1_green, and save the difference image.
Code: implement difference_image()
Report: The difference image as ps1-4-d-1.png in write up (make sure that the values are proper, e.g., do not exceed the limits of the image, when you write the image so that you can see all relative differences)

1.e. *Noise

a. Using Image1, start adding Gaussian noise to the pixels in the green channel. Increase sigma until the noise is visible but doesn't overwhelm the image. The full RGB image should be visible, just with some added noise in the green channel.
Code: implement add_noise(), modifying sigma in experiment.py to create visible noise.
Report: The noisy green channel image as ps1-5-a-1.png in writeup.

b. Apply that same amount of noise to the blue channel, and observe how it affects the image.
Report: The noisy blue channel image as ps1-5-b-1.png in writeup.

1.f. *Discussion

Report: Answer the questions below in the writeup. Please do NOT use more than 1 slide and max 100 words to answer these questions. If you need more than 1 slide, you're probably not explaining correctly.

a. Use the image southafricaflagface.png and look at all three channels individually as monochrome.
i. Between all color channels, which channel most resembles a grayscale conversion of the original?
ii. Why is this?
iii. Does it matter if you use other images? (For this problem, you will have to read a bit on how the eye works/cameras to discover which channel is more prevalent and widely used)

b. What does it mean when an image has negative pixel values stored?
i. Why is it important to maintain negative pixel values?

c. In question 5, noise was added to the green channel and also to the blue channel.

i. Which one looks more noisy to you?
ii. Why?
iii.What sigma value was used to detect any discernible difference?

1.g. *Hybrid Images

a. Hybrid images are static images that change in interpretation as a function of the viewing distance. The basic idea is that high frequency tends to dominate perception when it is available, but, at a distance, only the low frequency (smooth) part of the signal can be seen. In this section we will implement a simplified version of the hybrid image creation algorithm proposed in the SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns.

A hybrid image is the sum of a low-pass filtered version of the one image and a high-pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff-frequency". In the starter code, the cutoff frequency is controlled by changing the standard deviation of the Gausian filter used in constructing the hybrid images.

Code: Complete the function build_hybrid_image().
a. **Report:** Tune the cutoff-frequency parameter and report scaled images that reflects the hybrid image change in perception.

b. **Report:** Explain how the cutoff-frequency impacts the final hybrid image.

Credits: Assignment developed based on a similar project by James Hays.