Problem Set 1

Problem 1

Everyone can be a photographer

As SuShi wrote in his poem, "It’ s a range viewed in face and peaks viewed from the side, assuming different shapes viewed from far and wide." This is a vivid description of mountains viewed from different directions. Similar phenomena are common in our daily life, where the same scene or object looks completely divergent in different photographs. From the lectures, we have learned that changing the viewpoint and focal length (extrinsic and intrinsic parameters) of the camera can produce completely opposite visual expressions and tell different stories. The following photos are examples that show the choices of lenses and viewpoints that alter the illustrations of the COVID era.

Your task is to take and **submit three groups of photos** inside Peking University. **Inside each group, various (2 or more) photos tell different stories by shooting the same object or scene with various camera settings.** You can describe each group of photos with one sentence of description. The class will vote on the best three groups of photos and post them on the website.

Note: Good photos take time to collect or create. Start earlier and select from a larger pool.

Problem 2

Camera parameters from the image

Straight roads and cubic buildings are representative of landscapes in Beijing. Walking along the street and taking a picture, you may have noticed that the parallel lines of the roads and buildings can be used to calculate the pose and intrinsics of the camera. Please take a photo using your mobile phone **wherever it is feasible to calculate the focal length and camera height** from the image. **Calculate the focal length and camera height of that image and submit the derivation process.**

Hint: Calculating the focal length relates to camera calibration with vanishing points; calculating camera height might need a reference object.

Note: You should specify how the coordinate systems of the world (global) and the camera (local) are set up. The camera should avoid operating in any mode other than the standard mode to minimize image distortion.

Problem 3

Image filtering and processing

This problem is intended to familiarize you with basic algorithms in Python, NumPy, and image filtering. This project requires you to implement six functions, each of which builds onto a previous function:

1. `cross_correlation_2d`
2. `convolve_2`
3. `gaussian_blur_kernal_2d`
4. `low_pass`
5. `high_pass`
6. `hybrid_image`

Image Filtering. Image filtering (or convolution) is a fundamental image processing tool. See chapter 3.2 of Szeliski and the lecture materials to learn about image filtering (specifically linear filtering). Numpy has numerous built-in and efficient functions to perform image filtering, but you are asked to write your own function from scratch for this assignment. More specifically, you will implement `cross_correlation_2d`, followed by `convolve_2d`, which would use `cross_correlation_2d`.

Gaussian Blur. As you have seen in the lectures, there are a few different ways to blur an image, for example, taking an unweighted average of the neighboring pixels. Gaussian blur is a special kind of *weighted* averaging of neighboring pixels. To implement Gaussian blur, you will implement a function `gaussian_blur_kernel_2d` that produces a kernel of a given *height* and *width*, which can then be passed to `convolve_2d` from above, along with an image, to produce a blurred version of the image.

High and Low Pass Filters. Recall that a low pass filter removes the fine details from an image (or, really, any *signal*), whereas a high pass filter only retails the fine details, and gets rid of the coarse details from an image. Thus, using **Gaussian blurring** as described above, implement `high_pass` and `low_pass` functions.

Hybrid Images. A hybrid image is the sum of a low-pass filtered version of the one image and a high-pass filtered version of the 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 paper, it is suggested to use two cutoff frequencies (one tuned for each image), and you are free to try that, as well. In the starter code, the cutoff frequency is controlled by changing the standard deviation (sigma) of the Gaussian filter used in constructing the hybrid images.

Submit your code (all six functions in hybrid.py), README, left_id.png, right_id.png, hybrid_id.png in one folder without subfolders. Fail to comply with the following requirements will result in a score of 0 as your score is automatically evaluated by a program!

Requirements:

1. Your code will be scored by directly running hybrid.py, which reads two images at a time for hybrid and saves the hybrid result as a new image.
2. **Please put all your code (hybrid.py) and images in one folder.** Make sure that all the image paths (input and output) in hybrid.py are relative paths, in a form that hybrid.py can correctly execute to produce hybrid images.
3. **We provide five pairs of aligned images that can be merged reasonably well into hybrid images.** We encourage you to create additional examples (e.g., change of expression, morph between different objects, change over time, etc.). See the **hybrid images project page** for inspiration.
4. In this assignment, you should not use Numpy, Scipy, OpenCV, or other pre-implemented functions that directly finish the task. You are allowed to use basic matrix operations like np.shape, np.zeros, and np.transpose. For high efficiency of the code, you are encouraged to make full use of Numpy vectorization and avoid nested for loops. You are free to call your own previous functions during implementation.

Submit your project with a compressed folder that contains directories: problem1, problem2, problem3. Put the result in the corresponding directory.