Homework 1

September 22, 2023

1 Short questions (15 pt.)

1.1 Homogeneous coordination (10 pt.)

In the Cartesian coordinate system, a point in N-dimensional space is represented using N coordinates $P = (x_1, x_2, \ldots, x_N)$. In homogeneous coordinates, we represent the same point using $N + 1$ coordinates $P' = (x_1, x_2, \ldots, x_N, w)$, where w is a non-zero real number representing the weight or scale of the point. For simplicity, we may rewrite P' as $(x'_1, x'_2, \ldots, x'_N, 1)$.

You are required to:

- **Find the mathematical relationship between x_i and x'_i. (3 pt.)**
- **Please elaborate on at least two advantages of homogeneous coordination, and why we adopt it in computational photography. (7 pt.)**

(Hint: Consider the calculation of translation, rotation, scaling in linear algebra.)

1.2 Dolly zoom (5 pt.)

A dolly zoom (also known as a Hitchcock shot or Vertigo shot) is an in-camera effect that appears to undermine normal visual perception. The dolly zoom technique can create visual effects where the background suddenly grows in size and detail and overwhelms the foreground, or the foreground becomes immense and dominates its previous setting, depending on how the dolly zoom is executed. Figure 1 shows a classic example of a dolly zoom effect. **Please analyze how this effect is achieved.**

(Hint: Consider camera movement and changes in the field of view (FoV). You can easily search for more gifs and short videos on the Internet.)

Figure 1: An example of dolly zoom.

2 Camera parameters from the image (25 pt.)

Multiple sets of parallel structures on the spatial horizontal plane can provide geometric information about the camera or scene, such as camera pose and object heights.

Figure 2 shows a picture of our campus with dimensions of 4096×3072 pixels. In this context, we derive a world coordinate system (right-handed system) with the ground as the $z = 0$ plane, the edge of the roadside lawn as the x-axis, and the edge below the flowerbed as the y-axis, all measured in meters (m). At the same time, we measure the height of the first flowerbed (red rectangle) as 0.76 meters, and the camera's x and y coordinates as $(13.4, 4.5)$. We also take the edge point A of the hexagonal flowerbed, with its position being $(0, 3.74, 0.76)$ in the world coordinate.

Figure 2: A figure of our campus. Your task is to determine the camera height and focal length using this image.

You are required to:

- **Find the vanishing line and calculate the camera height H. (10 pt.)**
- **Write down the intrinsic parameter matrix and derive the camera focal length f. (15 pt.)**

Please display your graphical traces and explicit calculation process.

(Hint: For question 1, please refer to Figure 3. For qustion 2, stablish the camera coordinate system, construct the external parameter matrices T and R, and utilize the projection equation to calculate f.)

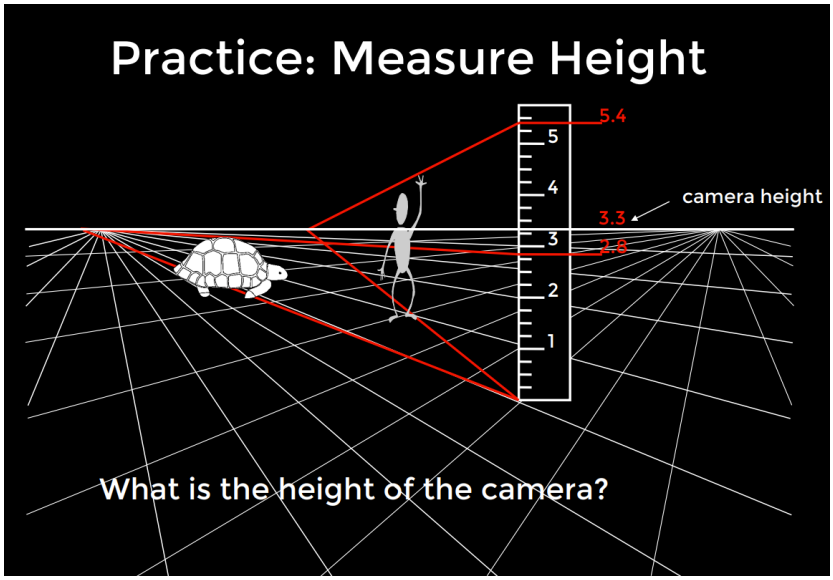

Figure 3: A visual explanation for question 1. The vanishing line indicates relative camera height, where you may use a reference object to compute the actual height.

3 Image filtering and subsampling (60 pt.)

This problem is designed to help you become familiar with basic algorithms in Python, NumPy, and image processing. This project requires you to implement six functions, each of which builds onto a previous function, except 3. and 5. :

1. `cross_correlation_2d`
2. `convolve_2`
3. `gaussian_blur_kernal_2d`
4. `low_pass`
5. `image_subsampling`
6. `gaussian_pyramid`

3.1 Image Filtering. (20 pt.)

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 Gaussian filtering). You are asked to write down your function from scratch for this assignment without using pre-implemented libraries. More specifically, you will implement `cross_correlation_2d`, followed by `convolve_2d`, which would use `cross_correlation_2d`.

3.2 Gaussian Blur. (10 pt.)

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 need to build 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.

3.3 Low Pass Filters. (10 pt.)

A low pass filter removes the fine details from an image (or, really, any signal) Thus, using Gaussian blurring as described above, implement `low_pass` functions.

3.4 Image subsampling/downsampling. (10 pt.)

Given an image with dimensions M, we subsample it by a factor of s, which results in an image with dimensions $(M/s) \times (N/s)$. Of course, s should be a common divisor of M and N. If we view the image as a matrix, each $s \times s$ window is compressed into a single pixel. In this problem, you are required to throw away every other row and column to create a 1/2 size image, like all even or odd rows and columns.

3.5 Gaussian pyramid. (20 pt.)

In a Gaussian pyramid, subsequent images are weighted down using a Gaussian blur and scaled down. Each pixel containing a local average corresponds to a neighborhood pixel on a lower level of the pyramid. In this problem, you have a free parameter that can be adjusted for each image to control the amount of high-frequency content removed from the image. You are required to generate a Gaussian pyramid with level 4: original image, 1/2 resolution, 1/4 resolution, 1/8 resolution.

Requirements:

1. Please prepare a report containing your answers for questions 1 and 2. For question 3, create a Python file named `gaussian_pyramid.py` and implement the above functions. We will evaluate your code by running `gaussian_pyramid.py` directly. It should read one image at a time and save the other three images (at 1/2 resolution, 1/4 resolution, and 1/8 resolution) as separate images.

2. Please put all your code (`gaussian_pyramid.py`) and images in one folder. Be sure that all the image paths (input and output) in `gaussian_pyramid.py` are relative paths, in a form that `gaussian_pyramid.py` correctly executes to produce a Gaussian pyramid.

3. We provide two images for the Gaussian pyramid, and we encourage you to create additional examples.

4. In this assignment, you Should Not use Numpy, Scipy, OpenCV, or other pre-implemented functions that directly finish the task. But you are allowed to use OpenCV, PIL, matplotlib, or other libraries only for reading and saving images. And you are also allowed to use basic matrix operations like `np.shape`, `np.zeros`, and `np.transpose`. For the 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 functions during implementation.