Homework 4

Computational Vision, Fall 2024

Due Date 11/11/2024

Total Points: 30

Please **follow the instructions** provided in the Programming Rules document on Piazza (under Resources). Note that failure to follow the rules will result in loss of points in the assignment. This is an individual assignment and all code and theory should be your work. You can use code I have provided. If you want to use any external source code, please consult me first. Please submit via GradeScope. Make sure that your code compiles and runs correctly in the cslab machines before submitting on Gradescope. Note that we will check your code for plagiarism using Gradescope. For the programming assignment submit only your source files (.cc and .h) as well a Makefile, README.txt, and thresholds.txt file. For this homework you should submit `s1.cc, s2.cc, s3.cc, image.h, image.cc, Makefile, README.txt and thresholds.txt`. Your `thresholds.txt` should contain one number, corresponding for the threshold used in program s3. The command `make all` in the terminal should generate all the executables listed below. Note that you should use the exact names of the executables as described below (s1, s2, and s3). To visualize the input/output images you can use the free software gimp (www.gimp.org)

(TOTAL: 30 points)

Theory Assignment (submit a pdf or scanned document via Gradescope)

A Lambertian surface is illuminated simultaneously by two distant point sources with equal intensity k (k is a scalar) in the directions $\mathbf{s_1}$ and $\mathbf{s_2}$ (the directions are unit vectors). Show that for all unit normals $\mathbf{n}$ on the surface that are visible to both sources, illumination can be viewed as coming from a single "effective" direction $\mathbf{s_3}$. In order to prove that you have to compute that "effective" direction.

(a) How is $\mathbf{s_3}$ related to $\mathbf{s_1}$ and $\mathbf{s_2}$?

(b) Now, if the two distant sources have unequal intensities k_1 and k_2, respectively, what is the direction and intensity of the "effective" source $\mathbf{s_3}$?

(Hint: Assume that the radiance due to source $\mathbf{s_1}$ is $L_1 = k_1 * \rho * (\mathbf{n} \cdot \mathbf{s_1})$ and the radiance due to source $\mathbf{s_2}$ is $L_2 = k_2 * \rho * (\mathbf{n} \cdot \mathbf{s_2})$, where ρ is the albedo of the surface, and k_1, k_2 are the intensities of the two sources. The operator $\cdot$ between the vectors is the dot product. In the first part of the question $k = k_1 = k_2$, while in the second $k_1 \neq k_2$.

Now, the combined result will be a radiance of $L_3 = L_1 + L_2$. Show that $L_3 = k_3 * \rho * (\mathbf{n} \cdot \mathbf{s_3})$, where k_3 is a scalar and $\mathbf{s_3}$ is a unit vector. The intensity of the combined source is k_3 and

the direction of the combined source is $\mathbf{s_3}$. You have to calculate k_3 and $\mathbf{s_3}$, from the known quantities $k_1, k_2, \mathbf{s_1}, \mathbf{s_2}$.)
(6 points)

Programming Assignment

In this programming assignment you are asked to develop a vision system that recovers the orientation and reflectance of an objects surface. For this purpose you will use photometric stereo. You will be given 3 images of an object taken using three different light sources. Your task is to compute the surface normals and albedo for this object. For this purpose, however, you will need to know the directions and intensities of the 3 light sources. Thus, in the first part of the assignment, you will compute the light sources directions and intensities from 4 images of a sphere and use this information in the second part to recover the orientation and reflectance. The 7 greyscale images, **sphere0.pgm ... sphere3.pgm and object1.pgm ... object3.pgm**, can be found on Piazza. Your program will be tested also with additional test images we have taken.
Before you begin, pay attention to the following assumptions you can make about the capture settings:

- The surface of all objects (including the sphere) is Lambertian. This means there is only a diffuse peak in the reflectance map (no specular component).
- For the images, assume orthographic projections.
- Image files with the same indices are taken using the same light source. For example, **sphere1.pgm** and **object1.pgm** are taken using light source number 1 only. The image **sphere0.pgm** is taken using ambient illumination.
- The objects maintain the same position, orientation and scale through the different images – the only difference is the light source. For example, the sphere in **sphere0.pgm ... sphere3.pgm** has the same coordinates and the same radius.
- The light sources are not in singular configuration, i.e. the S-matrix that you will compute should not be singular.
- You may NOT assume that the light sources are of equal intensities. This means that you need to recover not only the directions of the light sources but also their intensities.

The task is divided into three parts, each corresponding to a program you need to write and submit. Each program must accept arguments in the format specified as

program_name {*1st argument*} {*2nd argument*} ... {*Nth argument*}.

Program 1 First you need to find the location of the sphere and its radius. For this purpose you will use the image **sphere0**, which is taken using many light sources (so that the entire front hemisphere is visible).

Write a program s1 that locates the sphere in an image and computes its center and radius. The program parameters are as follows:

s1 {*input gray–level sphere image*} {*input threhsold value*} {*output parameters file*}

Assuming an orthographic projection, the sphere projects into a circle on the image plane. You need to threshold the greyscale image to obtain a binary one. Make sure you choose a good threshold, so that the circle in the resulting image looks clean. Then you can find the location of the circle by computing its centroid. As for the radius, you can average the differences between the leftmost and the rightmost and the uppermost and the lowermost points of the binary circle to obtain the diameter. Then divide by two to obtain the radius. The resulting parameters file is a text file consisting of a single line containing the
x-coordinate of the center, the y-coordinate of the center, and the radius of the circle, separated by a space.

(3 points)

Program 2 Now you need to compute the directions and intensities of the light sources. For this purpose you should use the images **sphere1.pgm ... sphere3.pgm**. We need to derive a formula to compute the normal vector to the spheres surface at a given point, knowing the points coordinates (in the image coordinate frame), and the coordinates of the center and the radius of the spheres projection onto the image plane (again, assume an orthonormal projection). This formula should give you the resulting normal vector in a 3-D coordinate system, originating at the spheres center, having its x-axis and y-axis parallel respectively to the x-axis and the y-axis of the image, and z-axis chosen such as to form an orthonormal right-hand coordinate system. Dont forget to include your formula in your README file. I will provide the formula in class.

Write a program **s2** that uses this formula, along with the parameters computed in **Program 1**, to find the normal to the brightest surface spot on the sphere in each of the 3 images. Assume that this is the direction of the corresponding light source (why is it safe to assume this?). Finally, for the intensity of the light source, use the magnitude (brightness) of the brightest pixel found in the corresponding image. Scale the direction vector so that its length equals this value.

Here are the program parameters:

s2 {*input parameters filename*} {*input sphere image 1 filename*} {*input sphere image 2 filename*} {*input sphere image 3 filename*} {*output directions filename*}

The input parameters file is the one computed in **Program 1**. The image files are the 3 images of the sphere (sphere 1 through 3, do not hardcode their names in the program!). The resulting directions file is a plain text file that consists of 3 lines. Line i contains the x-, y-, and z-components (separated by a space character) of the vector computed for light source i.

(7 points)

Program 3 Now you are ready to compute the surface normals of the object. Write a program **s3** that given 3 images of an object, computes the normals to that objects surface as well as the albedo at each point. You should use the formulas in the class lecture notes. Be careful here! Make sure to take into account the different intensities of the light sources.

Assume that a pixel (x, y) is visible from all 3 light sources if its brightness in all 3 images is greater than a certain **threshold**. Do not compute for pixels that are not visible in all images. The threshold is a parameter that will be supplied as a command line argument. So, for each pixel (x, y) with brighness greater than the **threshold** in all three images, compute the normal and albedo. Do not display normals for each pixel of the object. Display only for a grid of pixels, i.e. display the normals every N pixels along x and y axes. The value of N will be supplied as another command line argument, named **step**.

The programs parameters are as follows:

s3 {*input directions filename*} {*input object image 1 filename*} {*input object image 2 filename*} {*input object image 3 filename*} {*input step parameter (integer greater than 0)*} {*input threshold parameter (integer greater than 0)*} {*output normals image filename*} {*output albedo image filename*}

Here, the input directions file is the file generated by **s2**. The 3 image files are the files of the object taken with light sources 1, 2, and 3, in this order. The step and threshold parameters were described above.

The **output normals image** is input object image 1 with the normals "superimposed" shown as a "needle map". Show the gridpoints in black (use value 0). You may want to draw a small circle (of radius 1 pixel) around each gridpoint to make it easily visible.

From each grid point draw a line (a "needle") that is the projection of the normal vector at this point onto the image plane (again, use an orthographic projection). To make sure that this line is visible, scale the vector 10 times, after you normalize it (if it is normalized, the maximum projection will be 1 pixel, which is not very informative). Draw the needles in white color (255).

The **output albedo image** contains the albedos. Compute the albedos for all pixels visible from all 3 light sources, scale them up or down to fit in the range 0...255 and show them in the output image. Thus each pixel in the output image should be this pixels albedo scaled by a constant factor. Note that for pixels that an albedo cannot be calculated (see discussion about **threshold** above), just display black.

(14 points)