# Homework 5
# Computer Vision, Spring 2019
# Due Date: April 3, 2019
# Total Points: 14

This homework contains one written assignment and one programming challenge. All submissions are due at the beginning of class on **April 3, 2019**. The written assignment should be turned in as a hard copy at the beginning of class, and the challenge should be submitted according to the instructions in the document "**Guidelines for Programming Assignments.pdf**" before the beginning of class.

## Written Assignments

**Problem 1:** A Lambertian surface is illuminated simultaneously by two distant point sources with equal intensity in the direction $s_1$ and $s_2$. Show that for all normals on the surface that are visible to both sources, illumination can be viewed as coming from a single "effective" direction $s_3$. How is $s_3$ related to $s_1$ and $s_2$? Now, if the two distant sources have unequal intensities $I_1$ and $I_2$, respectively, what is the direction and intensity of the "effective" source? **(4 Points)**

## Programming Assignments

This programming assignment has one challenge (with its own subset of milestones or unit tests). Instructions and summary corresponding to these are given below. **runHw5.m** will be your main interface for executing and testing your code. Parameters for the different programs or unit tests can also be set in that file.

Before submission, make sure you can run all your programs with the command `runHw5('all')` with no errors.

**MATLAB is optimized for operations involving matrices and vectors. Avoid using loops (e.g., for, while) in MATLAB whenever possible—looping can**

**result in long running code. Instead, you should "vectorize** [1]**" loops to optimize your code for performance. In many cases, vectorization also results in more compact code (fewer lines to write!). If you are new to MATLAB, refer to these articles** [1] [2] **on techniques to optimize MATLAB code.**

**Challenge 1:** Your task is to develop a vision system that recovers the shape, surface normal and reflectance of an object. For this purpose you will use photometric stereo.

You will be given 5 images of an object taken using five different light sources. Your task is to compute the surface normal and albedo for the object. For this purpose, however, you will need to know the directions and intensities of the five light sources. Thus, in the first part of the assignment, you will compute the light sources directions and intensities from 5 images of a sphere and use this information in the second part to recover the orientation and reflectance.

The 11 images, **sphere0…sphere5**, and **vase1…vase5** are provided to you.

Before you begin, pay attention to the following assumptions you can make about the capture settings:

- The surfaces of all objects (including the sphere) are Lambertian. This means there are only diffuse peaks in the reflectance maps (no specular components).
- For the images, assume orthographic projections.
- Image files with the same indices are taken using the same light source. For example, **sphere1** and **vase1** are taken using light source number 1 only.
- 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…sphere5** 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 background in the image is black (0 pixel value) in all images.

The task is divided into four parts, each corresponding to a program you need to write and submit.

a. First you need to find the locations 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 `findSphere` that locates the sphere in an image and computes its center and radius.
   `[center, radius] = findSphere(input_img)`
   Assuming an orthographic project, the sphere projects into a circle on the image plane. Find the location of the circle by computing its centroid. Also estimate the area of the circle and from this, compute the radius of the circle.

   **You may use the Image Processing Toolbox functions `im2bw` and `regionprops`.**

   **(1 points)**

b. Now you need to compute the directions and intensities of the light sources. For this purpose you should use the images **sphere1…sphere5**.

   Derive a formula to compute the normal vector to the sphere's surface at a given point, knowing the point's coordinates (in the image coordinate frame), and the center and radius of the sphere's projection onto the image plane (again, assume an orthographic projection). This formula should give you the resulting normal vector in a 3-D coordinate system, originating at the sphere's 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. Don't forget to include your formula in your README file.

   Write a program `computeLightDirections` that uses this formula, along with the parameters computed in (a), to find the normal to the brightest surface spot on the sphere in each of the 5 images. Assume that this is the direction of the corresponding light source (Why is it safe to assume this? State this in the README).

   Finally, for the intensity of the light source, use the magnitude (brightness) of the brightness pixel found in the corresponding image. Scale the direction vector so that its length equals this value.

   ```
   light_dirs_5x3 = computeLightDirections(center, radius,
   ```

`img_cell)`

`Center` and `radius` are the resulted computed in (a). `img_cell` contains the 5 images of the sphere. The resulting `light_dirs_5x3` is a 5x3 matrix. Row i of `light_dirs_5x3` contains the x-, y-, and z-components of the vector computed for light source i. **(2 points)**

c. Write a program `computeMask` to compute a binary foreground mask for the object. A pixel in the mask has a value 1 if it belongs to the object and 0 if it belongs to the background. Distinguishing between the foreground and background is simple: if a pixel is zero in all 5 images, then it is background. Otherwise, it is foreground.
`mask = computeMask(img_cell)`
The `img_cell` contains the 5 images of an object and mask is the binary image mask. **(1 points)**

d. Write a program `computeNormals` that, given 5 images of an object, computes the normals and albedo to that object's surface.
`[normal, albedo_img] = computeNormals(light_dirs, img_cell, mask)`
You may want to use the formula given in the class lecture notes. Be careful here! Make sure to take into account the different intensities of the light source.

Photometric stereo requires the object to be lit by at least 3 light sources. However, in our case, we have a total of 5 light sources. The lighting has been arranged in such a way that all visible surface points on an object are lit by at least 3 light sources. Therefore, while computing the surface normal at a point, choose the 3 light sources for which the point appears brightest. Be careful – choosing the wrong light sources will result in erroneous surface normal. (You may also decide to choose more than 3 light sources to compute the surface normal. This results in an over-determined linear system and can provide robust estimates. However, such a computation is not mandatory.)

Do not compute the surface normal for the background. You can assume that the surface normal in this region is looking toward the camera. Use the mask generated in the previous program to identify whether a given pixel corresponds to the object or the background.

Scale the albedo up or down to fit in the range 0…1 and show them in the

output image. Thus each pixel in the output image should be the pixel's albedo scaled by a constant factor. **(6 points)**

At this point you can use the outputs of your program to reconstruct the surface of the object. `demoSurfaceReconstruction` and `reconstructSurf.m` demonstrate how to use the Frankot-Chellappa algorithm to compute the 3D shape from surface normals. You can use the MATLAB function `surf` to visualize the reconstructed surface. Surface reconstruction is provided as a demo--no submission is required.

## References


[1] MathWorks. Vectorization. [Online]. http://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

[2] MathWorks. Technique for Improving Performance. [Online]. http://www.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html

[3] Lytro Image Gallery. [Online]. https://pictures.lytro.com/

[4] Columbia CAVE. Focal Sweep Photography. [Online]. http://www.focalsweep.com/

[5] MathWorks. File Operations. [Online]. http://www.mathworks.com/help/matlab/managing-files.html