Homework 2
M1522.001000 Computer Vision (2016 Spring)
Due: Monday April 11 11:59PM

In this assignment we will explore the idea of photometric stereo and other properties of the image of a three-dimensional sense (i.e. object's surface properties, position, intensity and color of light sources).

1 Submitting Your Assignment

Your submission for this assignment will comprise of answers to two theory questions, the code for your MATLAB implementation and short writeup questions through 1 to 3 and interesting observations you made or things you did differently while implementing the assignment.

You need to zip the following items, name it as `M1522001000_HW2_(your student ID)#.zip`, and send it to TA's email.

Your submitted zip file should include the files arranged in this layout:

- `theory.txt` or `theory.pdf`
- Folder `result`
  - *bottle.fig*, *velvet.fig*, *wavy.fig*
  - `writeup.doc` or (or `.pdf`): the writeup describing your experiments
- Folder `matlab`
  - `createObjectVectors.m`, `PhotometricStereo.m`
  - You can also include any extra helper functions.

Refer to Section 3 for the details of the above files.

Your zip file should be sent before midnight of the due date. Later than that, you will use one late day.

2 Theory Questions

Question 1: Combining Light Sources (10 points)

A Lambertian surface is illuminated simultaneously by two distant point sources with equal intensity in the directions 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?

Question 2: Computing Scene Normals (10 points)

Consider an image of a sphere formed under orthographic projection. Let the center of the sphere be the point (a, b) in image coordinates and let the radius of the sphere's image be R pixels. Derive a formula for the normal direction to the sphere's at any point (u, v) on the sphere's surface (specified in image coordinates). The formula should give the normal vector in a 3D coordinate system with origin at the sphere's center and with x and y axes oriented with the image axes.

3 Programming & Writeup Questions

Reconstructing the shape of an object is an interesting and sometimes challenging task that generally requires more information than what is available from a single image. The simplest approach is to take three shaded images rather than one, but vary the position of the light source. This is photometric stereo. In this section we will implement some interesting (and simpler) parts of the following paper:

`Hertzmann, Aaron, and Steven M. Seitz. "Example-based photometric stereo: Shape reconstruction with general, varying brdfs." Pattern Analysis and Machine Intelligence, IEEE Transactions on 27.8 (2005): 1254-1264.`

Although we will not be implementing the entire paper, it is recommended that you read the paper to gain an understanding of the approach discussed in the paper. We will explicitly indicate which sections of the paper you need to read and implement. Please read and understand the sections prior to implementing the sections in code, so that you can avoid much of the troubleshooting issues.

3.1 Object Vectors (20 Points)

In this section you will implement *Section 2: Shape by Example*. Read *Section 2*.

Q1) Describe in your own words the "Orientation-consistency cue", and how it could help shape reconstruction? (2 sentences)
Q2) How does having more source images help improve the quality of the recovered shape? (2 sentences)

From reading the section, you should understand the general process that the authors use to recover the geometry of the unknown object. Begin by implementing the function:
`[ObjVec] = createObjectVectors(ImageName, NumberOfImages, ColorIndex)`
With inputs:

- `ImageName` - The *seed* name of the sequence of images as a string (the corresponding images are assumed to be in the current directory or in matlab's path). Use "bottle" for an image sequence `bottle1.png`, `bottle2.png`, ... and "sphere" for the image sequence `sphere1.png`, `sphere2.png`, ... etc.
- `NumberOfImages` - The number of images in the image sequence. If the images sequence contains images `bottle1.png` to `bottle8.png` then this number is 8.

- `ColorIndex` - the color index for which you are calculating the object vector. `ColorIndex=1` corresponds to "Red"–`Image(:,:,1)`, `ColorIndex=2` corresponds to "Green"–`Image(:,:,2)` and `ColorIndex=3` corresponds to "Blue"–`Image(:,:,3)`.

You should basically use the definition of the object vector within the paper to construct the `ObjVec`. The output `ObjVec` should be a matrix `MxN`, where `M` is the number of source images (i.e. `NumberOfImages`) and `N` is the number of pixels in one source image. Therefore, each column on the matrix represents the object vector of a pixel in the images (the same pixel over several source images). Note that the object vectors that you are creating are only for a single color value (i.e. it is not for RGB images and so you would need to calculate three sets of object vectors for a color image). You should also keep in mind that the object vectors that you have created here, are for every pixel in the image and not just the object of interest (i.e. it includes the background as well). Properly segmenting the image, to separate the object of interest and the background will greatly reduce the computation times. We'll address this problem in detail in the next section.

3.2 Matching and Surface Normals (30 Points)

Re-read *Section 2.1.1* and understand how they solve for the normals of the target image. Once you have calculated the object vectors for the reference and target images, let us proceed to the problem of matching each target object vector with its corresponding "closest-match" object vector in the set of reference object vectors. To do this, expensive search, we have provided the library for a kd-tree search algorithm. This library is a mex-file, which needs to be compiled within matlab. The process of compiling the library and instructions for using the library are available within the "README.txt" file in the "kdtree" directory. Please take the time to understand how the library works so that you can fully make use of the library's capability.

Once you have compiled the library, you can use the function `IDX = KDTREEIDX2(REF_OBJVEC,MODEL_OBJVEC)` to perform an efficient search of the object vectors. The inputs `REF_OBJVEC` and `MODEL_OBJVEC` should be the object vectors corresponding to the reference and the target images. The output of this function is a list of indices (`IDX`), which specifies the index of the reference object vector that corresponds to the appropriate target object vector. Feel free to use type `"help kdtreeidx2"` to better understand the use of the function.

Now that you have compiled the kdtree search algorithm we can now write a wrapper script that implements the core of the algorithm described in *Section 2* of the paper. Write the function:
`PhotometricStereo(TargetImageName, RefImageName, NumberOfImages, isRefASphere)`
that performs the following:

- The inputs to the function are as follows:
  - `TargetImageName` - is initialized with the string that specifies the *seed* name of the sequence of images for the target object.
  - `RefImageName` - is initialized with the string that specifies the *seed* name of the sequence of images for the reference object.

  - `NumberOfImages` - is initialized with the number of images in sequence of images for both the target image and the reference image.
  - `isRefASphere` - is initialized with '1' if the reference image is a sphere and '0' if the reference image is cylinder.

- We have provided you with three directories "bottle", "velvet" and "wavy", each with a set of reference images (in all cases the reference images are spheres or cylinders), and a set of target images. Each directory also contains a "README.txt" file, which contains brief description of the files within that folder, Now use the `createObjectVectors` function to generate the object vectors for both the reference images and the target images within one of the three directories mentioned above.

- Use the object vectors for the reference and target objects to perform matching using the `kdtreeidx` function. Since the object vectors that you have created include the pixels that do not belong to the objects of interest in each image, we must segment the image such that we only match object vectors that belong to pixels representing the objects (and not the backgrounds). To help you do this, each directory also contains a set of image "masks", which specify the pixels that belong to the object within the image. Using these masks you can easily segment the image into the object and the background, thus ensuring a more reliable match between the object vectors. The "README.txt" file contains more details about the "masks" for each image (reference and target).

- Use the known geometry of the reference object to calculate the surface normal for each pixel in the target object from the matching pixels in the reference object. Since only two types of reference objects are provided (sphere and cylinder), perform a simple check using the `isRefASphere` input variable to determine which reference object is provided. The surface normal that you calculate should be of the form of a matrix. This matrix, let's call it `Normal`, has dimensions `MxNx3`, where `MxN` is the size of the target images. The matrix, `Normal`, contains the vector (n_x, n_y, n_z), the normal vector, for each pixel along its third dimension.

- Use the surface normal matrix (`Normal`) that you computed as input for the function `[Ni,Z] = integrability2(Normal,[], nrows,ncols)`, which we have provided for you, to compute the surface `Z` for the target object. The inputs for this function include the surface normal matrix (`Normal`) that you have computed, an empty matrix (`[]`), the last two inputs (`nrows` and `ncols`) should be power of 2 (i.e. 2^7, 2^8, ..., 2^{11} - do not use anything more that 2^{11} since it will max out many systems). The inputs `nrows` and `ncols` should be larger than the horizontal and vertical dimensions of `Normal` (the larger these values, the slower it is but the more accurate the results will be). The outputs from this function are `Ni` - an integrable set of normal vectors (the same size as `N`), and `Z` - the surface depth values of dimensions `MxN` for a `MxNx3 Normal` matrix.

- The variable `Z` can now be used with matlab's built-in functions `mesh` and `surf` to plot the three dimensional surface of the target object. Make use of other built-in

functions, such as `colormap`, `light`, `axis equal` or `axis normal`, to help visualize the surface correctly. Save the images as a *.fig* file and submit these as part of the electronic submission for this assignment. Name these files using the seed name of each image sequence (i.e. `TargetImageName`.*fig* - *bottle.fig*, *velvet.fig* and *wavy.fig*).

Q3) Describe briefly the steps you followed to compute the surface normal matrix, *Normal*, in the function above? (2-3 sentences)

Be sure not to panic if the results that you acquire from the above implementation does not produce shape reconstruction that is as accurate as the ones presented in the paper. Due to the need for a practical implementation and the high computational time we had to reduce the size of the input images. This in turn introduces the level of inaccuracy that you observe.