Three-dimensional shape reconstruction using photometric stereo

Goal:

The goal of this project is to introduce a basic method for reconstruction of a three-dimensional surface using the concept of photometric stereo. We will see that it is possible to reconstruct the underlying shape of an object using only shading information (brightness or intensities of an image)

Description:

Image formation:

We will assume that the surface is Lambertian and that it is illuminated at each image by a given single light source. Lambertian surface means that a surface point has the same apparent brightness from any viewing angle. At image location (x,y) the intensity of a pixel I(x,y) is given by

$$I(x,y) = \rho(x,y)\ \vec{n}\ (x,y)^T\ \vec{s}$$

where:

$\rho(x,y)$ is the albedo (the extent to which the surface reflects light) at each point (a scalar)

$\vec{s}$ is a three-dimensional vector of a relatively distant light source (the same for the whole image). ($\vec{s}$ includes the direction and the intensity of the light source.)

$\vec{n}(x,y)$ is a three-dimensional vector representing the direction of the surface normal at each point

3D reconstruction from 3 images

The input images are images of the same scene taken from the same camera view but under different lighting conditions.

When we are interested in finding the three-dimensional shape of the imaged surface, we basically would like to recover the surface normals $\vec{n}(x,y)$ and integrate them to get the surface $z(x,y)$ (depth map). Since in our case the light will be given, we can recover $\vec{b}(x,y) = \rho(x,y)\vec{n}(x,y)$ from as few as three images. For each pixel we have

$$[I_1(x,y) \quad I_2(x,y) \quad I_3(x,y)] = \vec{b}(x,y)^T [\ \overrightarrow{s_1} \quad \overrightarrow{s_2} \quad \overrightarrow{s_3}]$$

here $\vec{s_i}$ is the light source (direction and intensity) of image $I_i(x, y)$.

(Note that because all 3 images were imaged from the same viewpoint, there is no correspondence problem, i.e., corresponding pixels are at the same image coordinates in all 3 images.)

We can solve for the scaled (by albedo) surface normal $\vec{b}(x, y)$ by inverting the 3x3 matrix $[\vec{s_1} \quad \vec{s_2} \quad \vec{s_3}]$ i.e.

$$\vec{b}(x, y)^T = [I_1(x, y) \quad I_2(x, y) \quad I_3(x, y)] \, [\vec{s_1} \quad \vec{s_2} \quad \vec{s_3}]^{-1}$$

The surface normal is then $\vec{n}(x, y) = \vec{b}(x, y)/||\vec{b}(x, y)||$ and albedo $\rho(x, y) = ||\vec{b}(x, y)||$ (where $||.||$ represents the L2 norm).

3D reconstruction using more than 3 images:

In case we have more than 3 images available we can use all of them and solve the problem by least-squares. The equation becomes

$$[I_1(x, y) \quad \dots \quad I_n(x, y)] = \vec{b}(x, y)^T [\vec{s}_1 \quad \dots \quad \vec{s}_n]$$

denote it by

$$M = b^T L$$

where M is 1 x n (intensities from the n images), b is 1 x 3, and L is 3 x n, the solution to b is then

$$b = (L^T L)^{-1} L^T M$$

As before, the surface normal is $\vec{n}(x, y) = \vec{b}(x, y)/||\vec{b}(x, y)||$ and albedo $\rho(x, y) = ||\vec{b}(x, y)||$ (where $||.||$ represents the L2 norm).

Note: for programming purposes, instead of finding 'b' for each of the pixels separately you can solve for all pixels simultaneously by constructing the following matrices:

i. Matrix of images M of size p x n (n is the number of images, and p is the number of pixels in each image)
ii. Matrix of lights L of size 3xn (3-dimensional vector of light for each of the images)

Integration of the normals to get surface depth

Please read this short explanation on how to transform surface normals to depth (taken from Basri, Jacobs and Kemelmacher , IJCV'07)

Details:

1. Your program will take in multiple images as input along with the light source direction for each image, and it will estimate the surface normals and the albedo. Then it will integrate the normals to get the surface depth.
2. Write a function called "reconstruct3d.m" that performs the following:
    a. Input:
        i. txt file with image names
        ii. txt file with the lights
    b. Output:
        i. Normals $\vec{n}(x,y)$
        ii. Albedo $\rho(x,y)$
        iii. Depth map $z(x,y)$
3. Apply your function to these sets of images: Set 1, Set 2, Set 3, Set 4 (in each set you will find the images, txt file with lights – each row corresponds to a different image, and txt file with the names of the images).
4. For the face datasets (those are real images taken from the YaleB database) you may reduce noise by convolving the image with a small Gaussian kernel (blurring) but this might not be necessary.

Notes:

1. In the face datasets you are given 10 images for each face. Try to run your reconstruction algorithm with less images, and different choices of images. How are the reconstruction results affected by this? Include your observations in the report.
2. For the integration procedure:
    - You will need to prepare a binary mask to indicate where in the images the object we want to reconstruct resides.
    - Try also to run the integration assuming that the object that we are imaging covers the entire image, mask = 1 for all pixels
    - Compare with/without mask results in your report.
3. Presentation of results:
    - To present the resulted depth map, you can use this function (use the mouse to rotate the shape, to see the profile, etc.)
    - To display normals, show each of the three components as an image $n_x(x,y), n_y(x,y), n_z(x,y)$ (use imagesc in Matlab)
    - To display the recovered albedo, use imshow
4. In your report include the following:
    - The input images

- The normals, depth, albedo (according to the display instructions)
- Commentary about any issues that arose, ways to improve your method, etc.

5. Here is an example of how your albedo $\rho(x, y)$ and depth map $z(x, y)$ should look like. This reconstruction was computed using one of the face datasets you were given. In this case the mask is rectangular centered on the face. Albedo is on the left and depth is on the right.

Good Luck!