CS6320: 3D Computer Vision: Photometric Stereo – An alternative Method

Wei Liu Xiaowei Shen

December 8, 2008

1 Introduction

In this project we use various images of same objects under different light source direction to recover the albedo and normal of the object, then contruct a 3D depth map using normals.

This is a student project by Dr. Steve Seitz on his year 2005 Computer vision class. The project page give the method to compute the normal, albedo and 3D depth map. As it was well documented and closed related to our 3D computer vision class, we decide to follow that project.

The method we use is closed related to the assignment two but there is also difference with it:

- The light source direction is unkown. In the assignment two the source direction is given, but in this project we need to caculate the direction of the source based on a 'chrome' image under the same light source condition. Because we know the shape of the sphere, we can compute the source light direction if we can compute the 'lightspot' of each chrome images.
- To get the normals of each pixels, we basically use the same method in assignment two. But for albedos, we can use another least square method once we get the normal for each pixel. As the original images are color images, we can get the albedos map for each R, G, and B channel.
- Once we get the normals of each pixels, we can use a method to recover the depth map different with assignment two. As the normal is orthogonal to the surface, it is perpendicular to any line/vector on the surface, including the vector from $(i, j, z(i, j))$ to $(i + 1, j, z(i + 1, j))$ and the vector from (i, j, z) to $(i, j + 1, z(i, j + 1))$. We can use this property to get a linear equation for the depth $z(i, j)$, and try to sove the over-constrained least square problem.

2 Detailed Algorithm

This section gave detailed information for computing the light direction, the normals and albedo, and the 3D surface of the object.

2.1 Bright spot of Sphere, by Wei Liu

To get the light direction from the 'chrome' image, we need first to know the bright spot of the sphere[2]. This is done by weighted averaging the pixels of the 'chrome' image. As there are same number of chrome image as the light source direction, we can get 10 bright spot from the 10 chrome images. When programming, only the pixels with intensity value > 0.9 are considered. This is to guarantee the noise are excluded from the computation. The code to get the brght spot can be found at: http://www.sci.utah.edu/~weiliu/cv/proj/getBrightSpotOnSphere.m

2.2 Get center and radius of the sphere, by Xiaowei Shen

In order to compute the normal of the sphere based on the x and y cordinates of the specular points, we need to get the raidus and sphere first. We can get this from the 'mask' image of the sphere. Basically the centroid is obtained by averaging the highest point and lowest point on the mask, and the radius is just half the difference between them. The code for doing this is under: http://www.sci.utah.edu/~weiliu/cv/proj/getCenterofSphere.m.

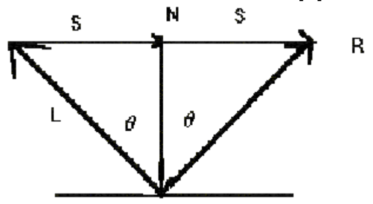
2.3 Light Source Direction, by Wei Liu and Xiawei Shen

For the method of getting light source direction, we actually refered the project's webpage and Phong's model for specular reflection.

as the image is taked over the object, we can assume the vector from the viewer to the specular point is perpendicular to the XY plane, so we can assume the vector from the specular point to the viewer is [0 0 1]

Also if we know the x and y coordinate of the specular point, we can compute its z coordinate of this point also, as the point is under constraint of sphere. With all teh x, y, z coordiantan easily compute the normal on this point.

Once we have the normal and the vector R = [0 0 1], we can use a geometry method to get the light direction. Look at the figure below [1]:



By the geometry of the figure we can get the light direction R as

$$\boldsymbol{R} = 2 \cdot \boldsymbol{N}(\boldsymbol{N} \cdot \boldsymbol{L}) - \boldsymbol{L}$$

Whre N is the normal of the point, and L is the viewer's direction ([0 0 1]). The code for doing this part is under: http://www.sci.utah.edu/~weiliu/cv/proj/getLightingDirection.m.

2.4 Get Normals and Albedos, by Wei Liu

With the direction of the light source available in previous section, we use the same method to compute the normal and albedo with the assignment 3[3] . The diffuse object can be modeled as

$$I = k_d \boldsymbol{L} \boldsymbol{n}^\top$$

Where k_d is the albedo, $\boldsymbol{L}$ is the light direction and $\boldsymbol{n}$ is the surface normal. I is the intensity (the pixel value). As k_d and $\boldsymbol{n}$ are both unknown, we combine them into one variable $\boldsymbol{g}$. So with 10 image under various light direction available, we can sove the least square problem

$$min(Q) = min(\sum_i (I_i - \boldsymbol{L}_i \boldsymbol{g})^2)$$

The albedo is computed on three channels R, G and B seperately. By the project page we can compute the albedo by soving another least square problem

$$Q = \sum_i (I_i - k_d \boldsymbol{L}_i \boldsymbol{n}^\top)^2$$

We need to get derivate with respec to albedo k_d and set it to zero we get the k_d

$$k_d = \frac{\sum_i I_i \boldsymbol{L}_i \boldsymbol{n}^\top}{\sum_i (\boldsymbol{L}_i \boldsymbol{n}^\top)^2}$$

The code for this part is under: `http://www.sci.utah.edu/~weiliu/cv/proj/getOneNorm.m`.

2.5 Get 3D surface map, by Xiaowei Shen

With the normals of the test image available, we can compute the 3D depth map. We use a different method with the assignment two. That is, instead of using gradient method, we consider the fact that any vector within the surface are perpendicular to the normal. The vector from $(i, j, z(i, j))$ to point $(i, j+1, z(i, j+1))$ is perpendicular to the normal of point (i, j). Similarly, the vector from $(i, j, z(i, j))$ to point $(i+1, j, z(i+1, j))$ is also perpendicular to the normal. Thus for the same point p we get two linear equation of the surface (the z coordinate). Th is is again a over-constrained system $\boldsymbol{M} \cdot \boldsymbol{P} = \boldsymbol{z}$. As M is extremely large (it have roughly $2N$ rows where N is the total number of the pixels under consideration), we have to use some methods like sparse matrix to store and compute this linear system.

The code for this part can be found at: `http://www.sci.utah.edu/~weiliu/cv/proj/make3Dsurface.m`.

3 results

This section gives a short introduction to the program usage and the results.

3.1 How to run the program

The main portal of running the progam is photometricStereo.m , which call all other functions to get light direction, normal and albedo and surface map of the test image. To run it, users need to copy the myimage.mat file to Matlab's current directory. This .mat file contains all the images used in this project.

To call the main program, simply use

```
photometricStereo (buddha, buddhamask)
```

All the files (including codes and figures can be found at: `http://www.sci.utah.edu/~weiliu/cv/proj/`.

3.2 Test on 'buddha' and 'owl' image

The original project provide 7 images for testing. We simply pick out the 'buddha' and 'owl' images. both 'buddha' and 'owl' have mask image to help decide the shape of the object, and ten images under various lighting direction for processing.

We first test the 'buddha' image. The bright spot we got from the ten 'chrome' images are

```
118.8608  286.1772
140.5738  268.9344
138.2174  251.9565
121.5733  248.4400
116.9014  234.1408
113.5862  247.3103
122.6625  271.6250
122.3837  260.4651
128.3243  266.9459
128.5352  259.6479
```

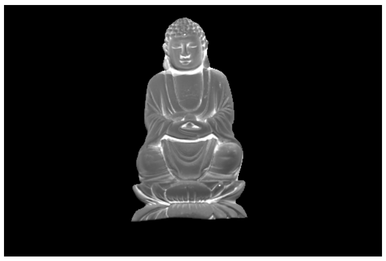
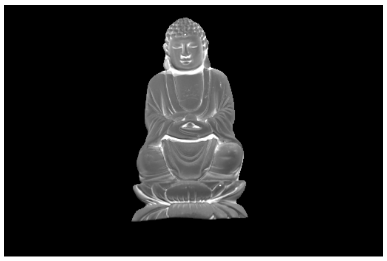
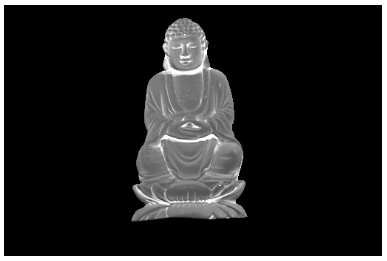
Each row contains the x and y coordinates (pixel positions) of the bright spot in one of the tem images.

The normalized ligh direction of the ten 'chrome' images are

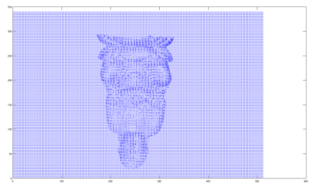
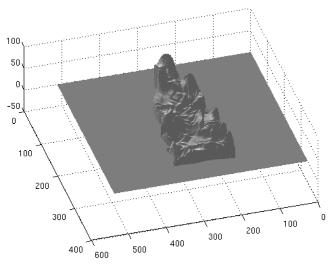
```
-0.3450    0.3683    0.8633
-0.0994    0.1762    0.9793
-0.1276   -0.0242    0.9915
-0.3207   -0.0650    0.9449
-0.3710   -0.2295    0.8998
-0.4103   -0.0775    0.9087
-0.3068    0.2053    0.9294
-0.3115    0.0757    0.9472
-0.2426    0.1519    0.9582
-0.2408    0.0664    0.9683
```

In the results above, we assume the center of the sphere is also the coordinate origin.

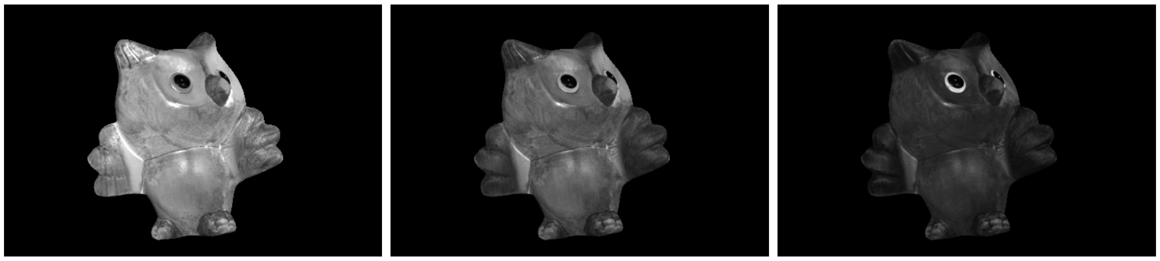
The R, G, and B albedo map for the 'buddha' object

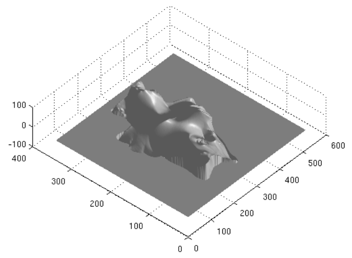
The normal's needle map and the 3D surface for the 'buddha' object:

for the 'wol' image, the R,G and B albedo map:



The 3D surface map for the 'owl' images are:



4 Analysis

We see by observing the 'chrome' images under same light condition, we can recover the light direction quite well. If the viewer is close to the 'chrome' object, the assumption that 'specular–¿ viewer' vector is perpendicular to the sphere surface may not hold, and that vector will not be a known [0 0 1]. But in our case, we can assume the [0 0 1] without introducing much error.

Basically the algothrim works except for some 'singular' places like the 'owl's eyss. That is because the intensity of this image blocks are almost zero, and it's hard to recover the normal information form pixel values. The depth map in this area is also undefined.

References

[1] Scott Owen. Methods to compute VR. 1999.

[2] Steve Seitz. Project 3: Photometric Stereo. 2005.

[3] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1), Jan/Feb 1980.