Assignment 1: Photometric Stereo

Jia Ming Liang
October 7, 2014

1 CALIBRATION

I have chosen the Phong model direct specular component to represent the system for the chrome sphere setup. The algorithm for my code:

1. **Calculate the center of the sphere**
   - We know that the mask is almost perfect circle so we can simply take the mean of the row and column index to find the estimated center.
   - Radius = $\frac{max[rowIndex]-min[rowIndex]}{2}$
2. **Find the brightest spot in the chrome sphere images.** To get that brightest pixel we assume that pixel is in the center of the shiny spot then we do the same thing as finding the center of the the mask (taking the mean of the row and column index). The bright pixel index can be found using the matlab code:
   $[pRow\ pCol] = find(maxpix == mask)$;
3. **Find the surface normal of the brightest spot on the chrome sphere.** Setting the center of sphere as origin:
   - Nx = Xpoint - Xcenter;
   - Ny = Ypoint - Ycenter;
   - Nz = $-\sqrt{R^2 - Nx^2 - Ny^2}$
     (We set this to negative because this is how we defined in our coordinate system)
   - Normalize N = [Nx Ny Nz] to get the unit vector

4. **Find the light source direction.** Based on the rule for specular refection:

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

where $\vec{R}$ = [0, 0, -1] and $\hat{N}$ = normalized surface normal at the bridght spot

Finally, here are the results:

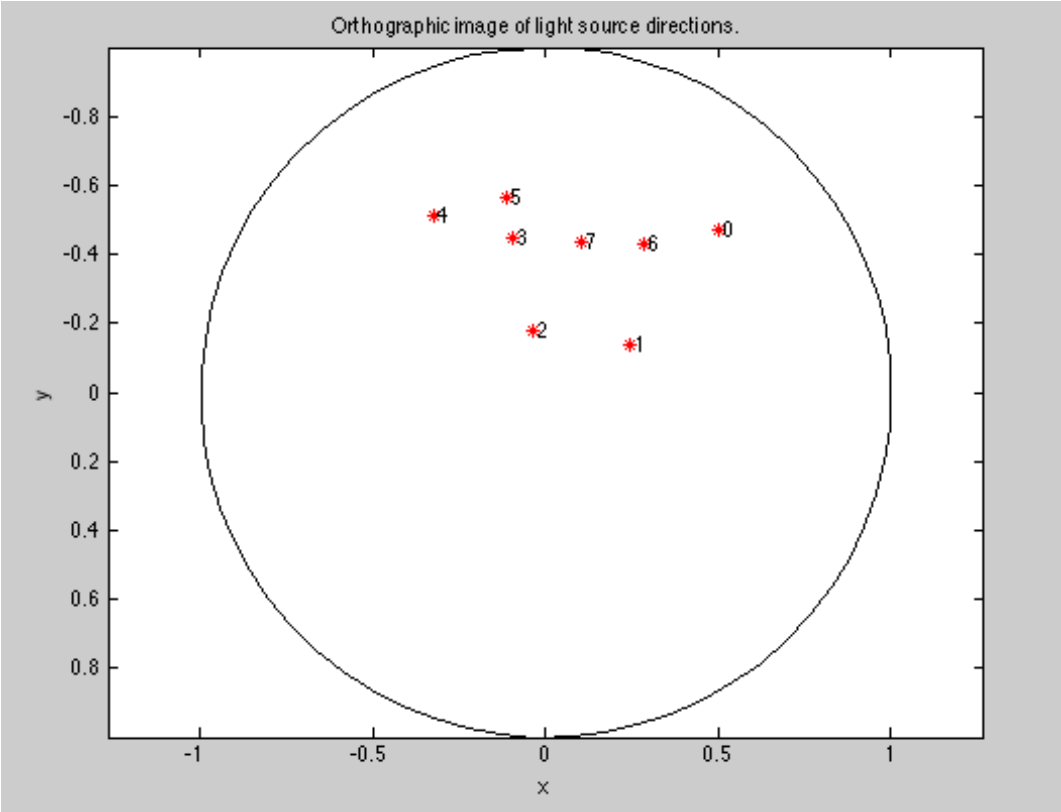

Figure 1.1: Recovered direction plot

2 Computing Surface Normals and Grey Albedo

To derive the formula to recover surface normal and grey albedo, we first differentiate E($\vec{x}$)

$$E(\vec{x}) = \sum_{j=1}^{8} (I_j(\vec{x}) - \vec{g}(\vec{x}) \cdot \vec{L_j})^2 \tag{2.1}$$

$$\frac{d}{d\vec{g}(\vec{x})}E(\vec{x}) = \sum_{j=1}^{8} \frac{d}{d\vec{g}(\vec{x})}(I_j(\vec{x}) - \vec{g}(\vec{x}) \cdot \vec{L_j})^2 \tag{2.2}$$

Since $\frac{d}{d\vec{g}(\vec{x})}E(\vec{x}) = 0$

$$0 = \sum_{j=1}^{8} \frac{d}{d\vec{g}(\vec{x})}(I_j(\vec{x})^2 - 2\vec{g}(\vec{x}) \cdot \vec{L_j} I_j(\vec{x}) + \vec{g}^2(\vec{x}) \cdot \vec{L_j}^2) \tag{2.3}$$

$$0 = \sum_{j=1}^{8} (-2\vec{L_j} I_j(\vec{x}) + 2\vec{g}(\vec{x}) \cdot \vec{L_j}^2) \tag{2.4}$$

For individual picture:

$$I(\vec{x}) = \vec{g}(\vec{x}) \cdot \vec{L} \tag{2.5}$$

Using equation (5.5) we can solve for $\vec{g}(\vec{x})$ because we were given $I(\vec{x})$ and we solved $\vec{L}$ in question 1. In matlab, I use least square method to solve for $\vec{g}(\vec{x})$ with the 8 images intensity (using the backslash g=L\I). After we solve $\vec{g}(\vec{x})$ for every single pixels, we can solve $a(\vec{x})$ and $\vec{n}(\vec{x})$. $a(\vec{x})$ is the magnitude of $\vec{g}(\vec{x})$ where $\vec{n}(\vec{x})$ is the unit vector of $\vec{g}(\vec{x})$.

Finally here are some results:

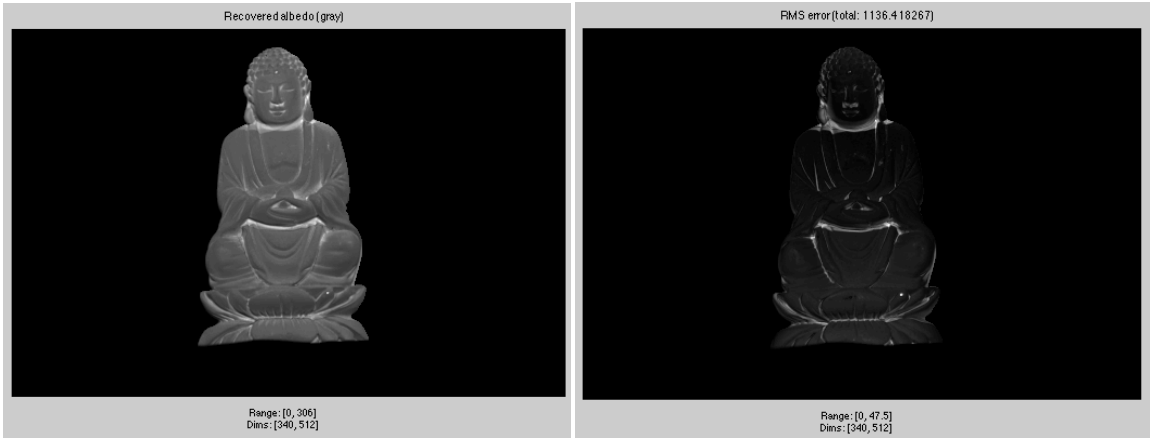

Figure 2.1: Buddha estimated grey albedo plot (left) and RMS error plot (right)

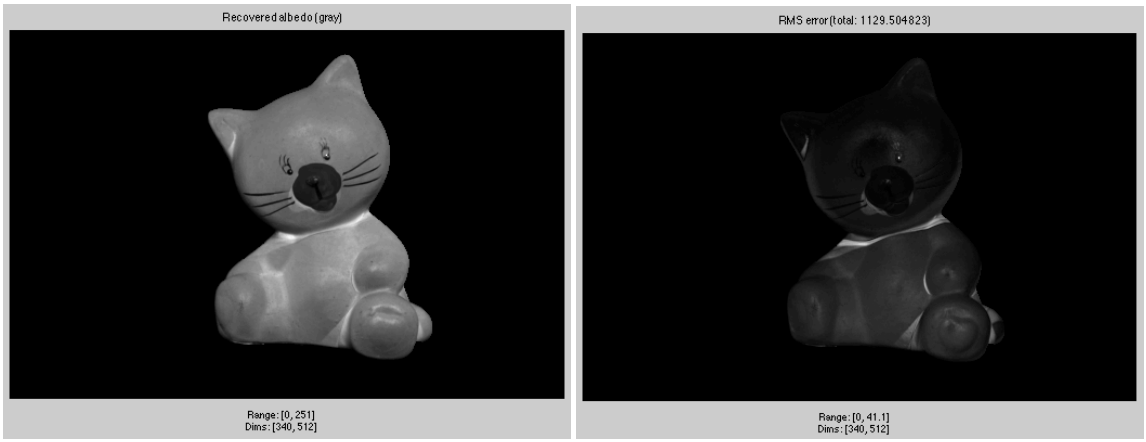

Figure 2.2: Cat estimated grey albedo plot (left) and RMS error plot (right)

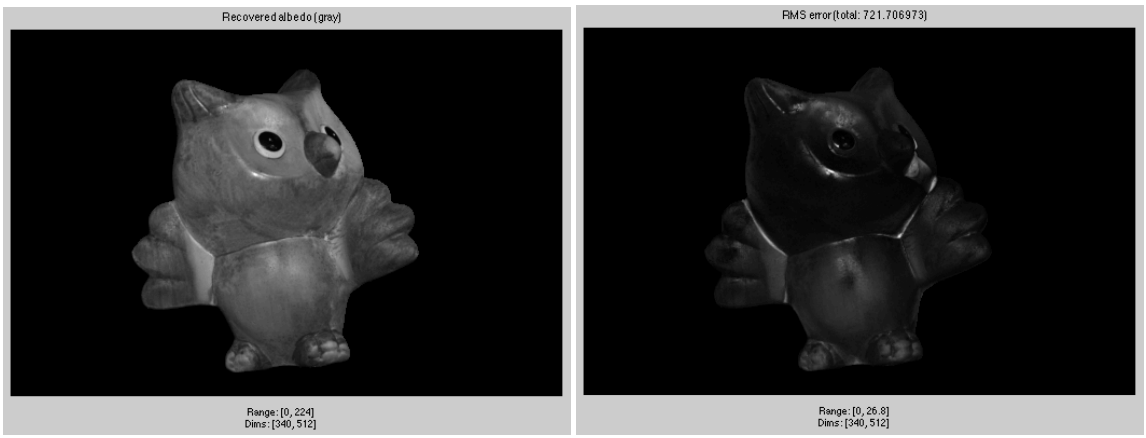

Figure 2.3: Owl estimated grey albedo plot (left) and RMS error plot (right)

The quality of the recovered grey-scale image look decent but there are area with high estimate error cause the image to lose realistic looks. Using the cat RMS error plot for example, the errors spike in the neck area and some area in the left leg and the right ear which seem to be in the same location as the shadow occurred in the original images (lower error if less images have shadow in those area). Our model assumes linear Lambertian reflectance. This model does not take into account of the shadow because it is expected to have light reflected directly off the surface of the object to the camera (with diffusive reflection). Also, there are light intensity spikes in the original image such as the bright spot below the right eyes of the owl and the tiny bright spot in the eyes of the owl and cat also have high error. This is because Lambertian surface are expected to be diffusive where those intensity spike spots are more like specular reflection (similar to the chrome ball with almost perfect reflection). This is because the original object have smooth

surface and the camera is positioned such that those spots are direct specular reflection. Because of this, those shadow area and shiny spots have higher estimated albedo value than it should have been.

Here are some surface normal plot (for individual components x, y and z):

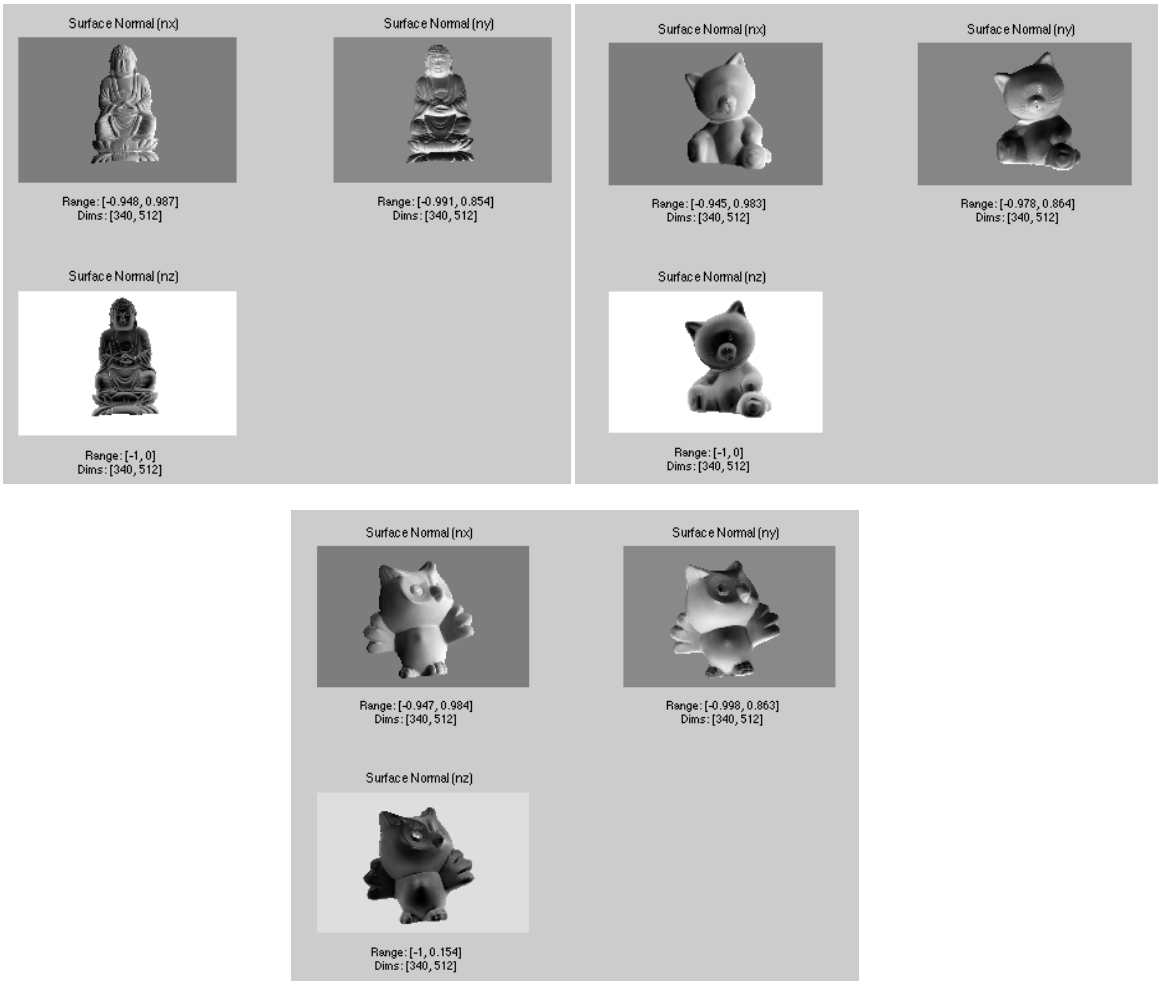

Figure 2.4: Recovered normal map plot

3 Computing Surface Normals and Grey Albedo

To minimize equation $E_v(\vec{x}) = \sum_{j=1}^{8} (I_v(\vec{x}) - a_v(\vec{x})\vec{n}(\vec{x}) \cdot \vec{L_j})^2$, we set $\frac{d}{da_v(\vec{x})} E_v(\vec{x}) = 0$

Similar to how we solve $\frac{d}{d\vec{g}(\vec{x})}E(\vec{x})$ in question 2

$$\frac{d}{da_v(\vec{x})}E(\vec{x}) = \sum_{j=1}^{8}\frac{d}{da_v(\vec{x})}(I_j(\vec{x}) - a_v(\vec{x})\vec{n}(\vec{x})\cdot\vec{L_j})^2 \tag{3.1}$$

Since $\frac{d}{da_v(\vec{x})}E(\vec{x}) = 0$

$$0 = \sum_{j=1}^{8}\frac{d}{da_v(\vec{x})}(I_j(\vec{x})^2 - 2a_v(\vec{x})\vec{n}(\vec{x})\cdot\vec{L_j}I_j(\vec{x}) + a_v^2(\vec{x})\vec{n}^2(\vec{x})\cdot\vec{L_j}^2) \tag{3.2}$$

$$0 = \sum_{j=1}^{8}(-2\vec{n}(\vec{x})\cdot\vec{L_j}I_j(\vec{x}) + 2a_v(\vec{x})\vec{n}^2(\vec{x})\cdot\vec{L_j}^2) \tag{3.3}$$

For individual picture:

$$I(\vec{x}) = a_v(\vec{x})\vec{n}(\vec{x})\cdot\vec{L} \tag{3.4}$$

As we can see equation (3.4) is same as equation (5.5). This time are we solving the individual color channel instead. So the algorithm used to solve $a_v(\vec{x})$ for all red, green and blue channel is the same as the one used in section 2. In section 2 we implemented the function
[nCrop, albedoCrop] = fitReflectance(imDataCrop, Lchrome);
The variable imDataCrop is the grey-scale intensity for every pixels in the masked region and Lchrome is the light source direction which we determined in section 1. All we have to do now is change the input imDataCrop to imDataRedCrop, imDataGreenCrop or imDataBlueCrop which can read directly from the orginal images. i.e code:
[~, albedoRGBCrop(:,1)] = fitReflectance(imDataRedCrop, Lchrome);
[~, albedoRGBCrop(:,2)] = fitReflectance(imDataGreenCrop, Lchrome);
[~, albedoRGBCrop(:,3)] = fitReflectance(imDataBlueCrop, Lchrome);
Since we do not need to calculate the surface anymore we simply ignore the normal output by replace nCrop with ~.
Below are some recovered albedo plots:

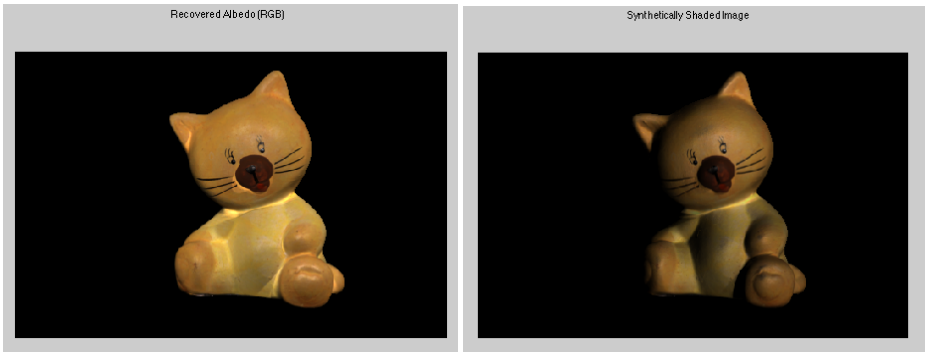

Figure 3.1: Recovered cat RGB albedo (left) and synthetically shaded image (right)

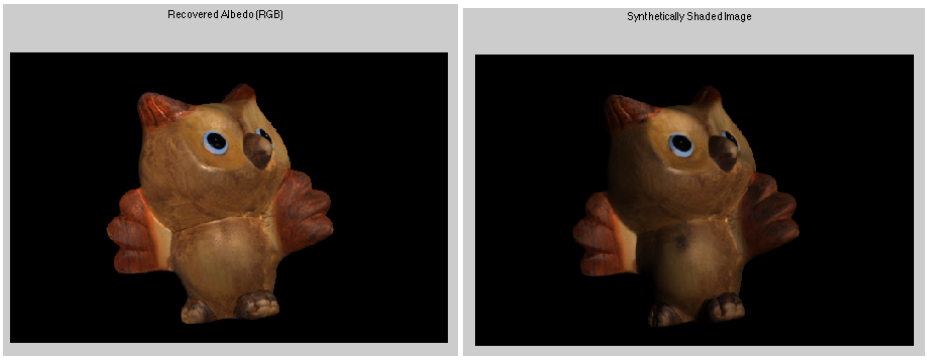

Figure 3.2: Recovered owl RGB albedo (left) and synthetically shaded image (right)

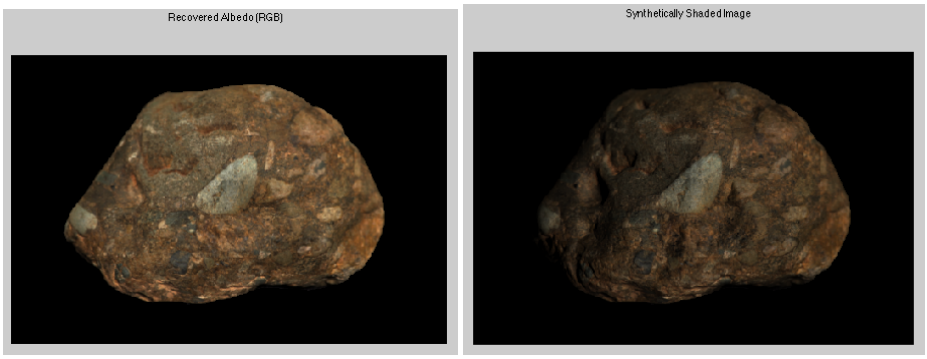

Figure 3.3: Recovered rock RGB albedo (left) and synthetically shaded image (right)

Just like we discussed in section 2, the quality of the newly synthesized images are decent except that it does not look realistic. If the albedos and surface normal are correct, the remainniing modelling errors in these synthetically light images are the specular reflection component and shadow. Again shadow and specular reflection are not accounted in our Lambertian surface model because it is purely diffusive reflection.

4 Surface Fitting

To find the depth map, we use the constrains given in the assignment 1 handout to form the sparse matrix **A** and $\vec{v}$. **A** is a large 2NM x NM matrix consist of mostly 0, every row will have 2 columns that have 1 and -1 (this represents $Z_{x,y} - Z_{x+1,y}$ or $Z_{x,y} - Z_{x,y+1}$) and the rest are 0. $\vec{v}$ will be a vector consist of 1 column and 2MN rows with the tangent values for the pixels. We then use these two to solve $\vec{z}$ which is given to be $\mathbf{A}\vec{z} = \vec{v}$ (in matlab, we use the least square method to

solve for z, sample code: z = A\v)

Putting together everything, it will look like

$$\begin{pmatrix} 1 & -1 & 0 & \cdots & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & -1 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{NM} \end{pmatrix} = \begin{pmatrix} \frac{N_{x1}}{N_{z1}} \\ \frac{N_{y1}}{N_{z1}} \\ \frac{N_{x2}}{N_{z2}} \\ \vdots \\ \frac{N_{yNM}}{N_{zNM}} \end{pmatrix}$$

z_1 to z_{NM} is the depth for every single pixel within the image mask region. My algorithm will create a NxM matrix with all 0 and using the mask to label every pixel with index from 1 to NM. I then use the pixel position to retrieve this index to place the correct 1 or -1 value in the proper column in the sparse matrix (please see the matlab code for clarification). For most cases, to form the constrain equations for pixel$_{x,y}$ I use pixel$_{x+1,y}$ and pixel$_{x,y+1}$. However, if the pixel$_{x+1,y}$ or pixel$_{x,y+1}$ is not within the mask then we simply use pixel$_{x-1,y}$ or pixel$_{x,y-1}$ (for this case we put negative sign in the tangent value).

Here are some results for the depth math:

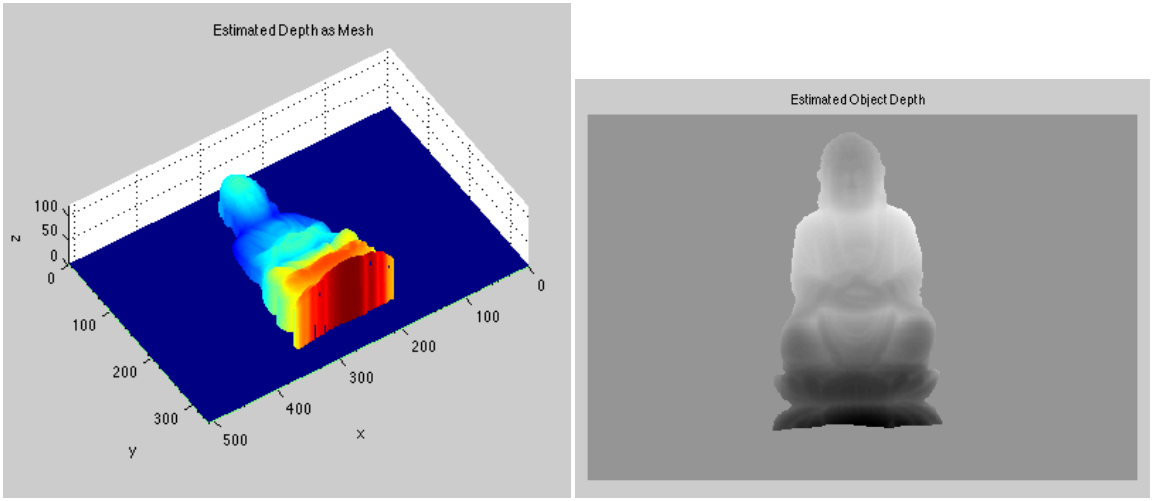

Figure 4.1: Buddha estimated depth mesh (left) and plot (right)

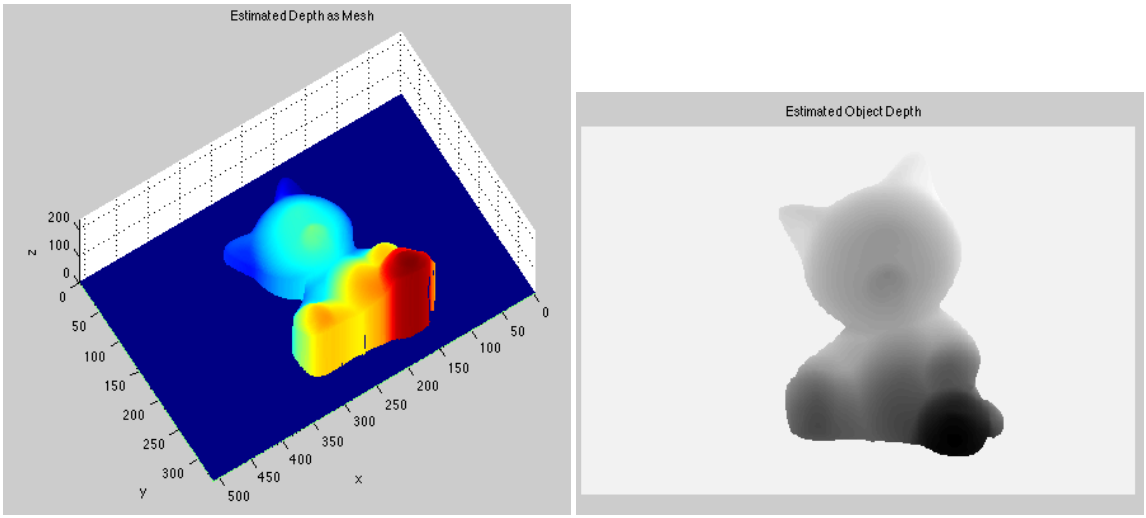

Figure 4.2: Cat estimated depth mesh (left) and plot (right)

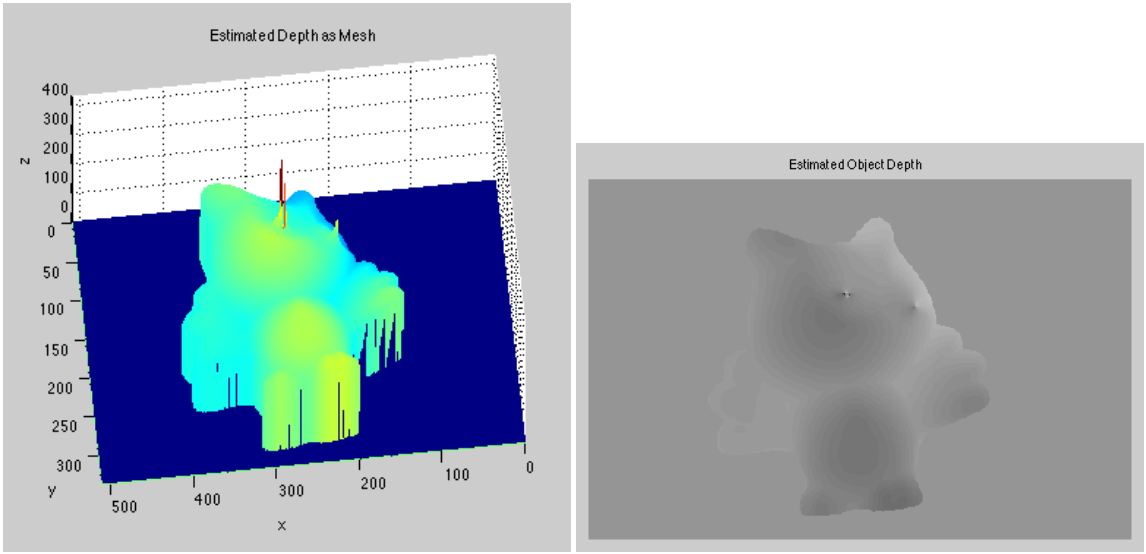

Figure 4.3: Owl estimated depth mesh (left) and plot (right)

In most area of the images the techique we used to retrieve the depth map work really well. We can see all 3 objects have restored their structure roughtly. Through the mesh map we can even see the original shape of the object. Compare the mesh map with the RMS error plot in section 2. We can see that the region with larger error seems to have some irregular change of contour in the mesh map. For example, the eyes of the owl have a few pixels that have sudden change in the tangent (see figure 4.4).

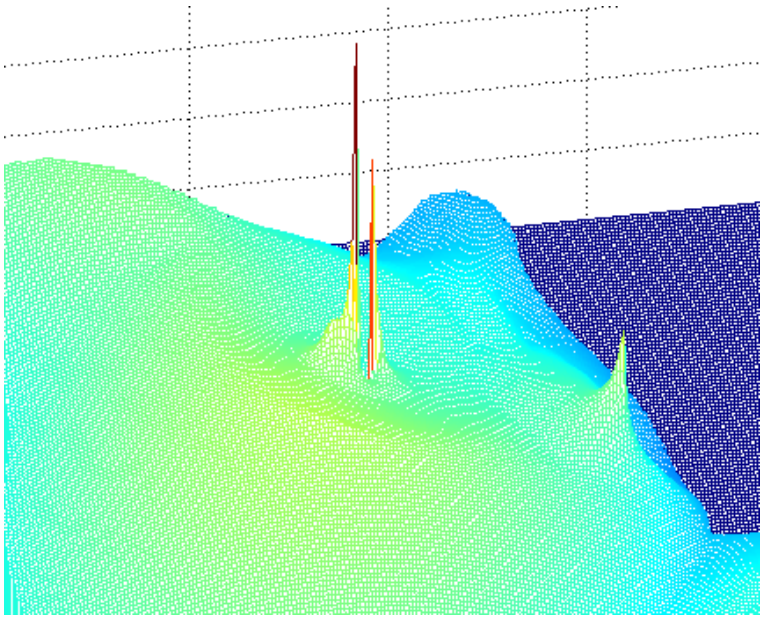

Figure 4.4: Owl estimated depth mesh closed up

In my algorithm, the tanget is determined by Nx/Nz. Large depth indicates that Nz is small. This spot seems to be where the specular reflection spot is. In our Lambertian surface model, our calculation of the surface normal do not account for specular reflect. Also, this specular spot is very tiny and the area around that spot is dark (because its inside the eyes). The bridgh spot may have very small Nz because of this (consistent with our RMS error plot). This tricks the algorithm into computing large depth difference.

The shadow area probably have error in the depth map too because the normal is incorrect. However, in the mesh plot we do not see it as obvious because it affects a larger area so the error is more continous as compare to the specular spot that has sudden brigthness change.

To fix these errors, we simply need to come up with a more accurate and complex model that account for shadow and specular reflection when we calculate the normal. Without change our model, we can also try to test the depth change in our object. If the depth chagne is too dramatic we can try to form the constrain with the pixels furthur away instead of right next to it. This could avoid the tiny spot with extreme brightness change. Or simply look at the result. If there are very tiny spots with huge depth difference we try to average it out with the pixels around that area.

5 Estimate Light Source Direction Given Scene Properties

Similar to section 2 and 3, we minimize the equation $E(\vec{x})$, this time with respect to $\vec{L}$

$$E(\vec{x}) = \sum_{j=1}^{8} (I_j(\vec{x}) - \vec{g}(\vec{x}) \cdot \vec{L_j})^2 \tag{5.1}$$

$$\frac{d}{d\vec{L}} E(\vec{x}) = \sum_{j=1}^{8} \frac{d}{d\vec{L}} (I_j(\vec{x}) - \vec{g}(\vec{x}) \cdot \vec{L_j})^2 \tag{5.2}$$

Since $\frac{d}{d\vec{L}} E(\vec{x}) = 0$

$$0 = \sum_{j=1}^{8} \frac{d}{d\vec{L}} (I_j(\vec{x})^2 - 2\vec{g}(\vec{x}) \cdot \vec{L_j} I_j(\vec{x}) + \vec{g}^2(\vec{x}) \cdot \vec{L_j}^2) \tag{5.3}$$

$$0 = \sum_{j=1}^{8} (-2\vec{g}(\vec{x}) I_j(\vec{x}) + 2\vec{g}^2(\vec{x}) \cdot \vec{L_j}) \tag{5.4}$$

For individual picture:

$$I(\vec{x}) = \vec{g}(\vec{x}) \cdot \vec{L} \tag{5.5}$$

The result is same as section 2. So similarly, we use the grey scale image intensity for $I(\vec{x})$ and use $a_v(\vec{x})$ and $\vec{n}(\vec{x})$ calculated in section 2 to solve for $\vec{L}$ with the least square method. Here is the snippet I added in matlab to solve for $\vec{L}$:

```
g = nCrop .* repmat(albedoCrop,1,3);
Lrec = zeros(3,nDir-nDirChrome);
imDataCrop = imData(mask,(nDirChrome+1):nDir);
for i=1:(nDir-nDirChrome)
    Lrec(:,i) = g\imDataCrop(:,i);
end
```

Here are the results of our estimation:

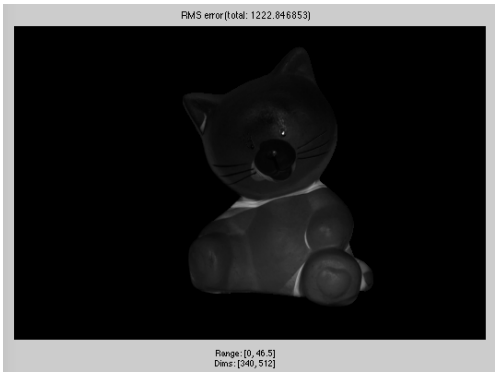

Figure 5.1: Cat RMS error plot for recovered light direction

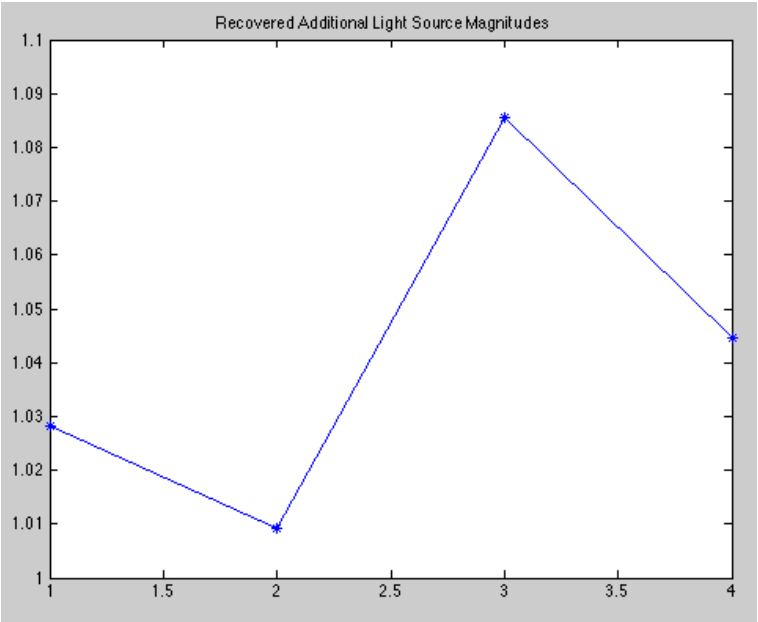

Figure 5.2: Orthographic image of light source directions plot

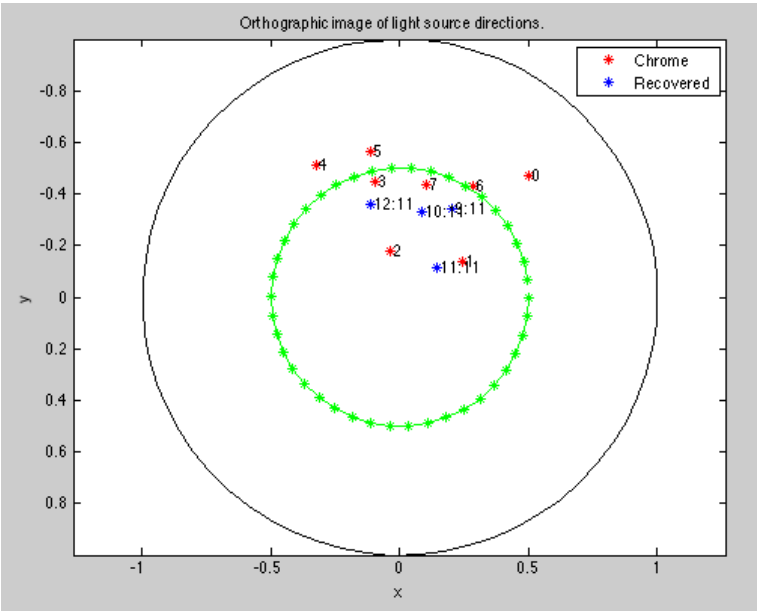

Figure 5.3: Orthographic image of light source directions plot

If we have two light source directions then our Lambertian surface model equation would be $I(\vec{x}) = \vec{g}(\vec{x}) \cdot (\vec{L_1} + \vec{L_2})$. With our method we can solve 3 numbers. Lets denote those 3 numbers to be A, B and C, then [A B C] = $\vec{L_1} + \vec{L_2}$. Both $\vec{L_1}$ and $\vec{L_2}$ have 3 unknowns (x, y and z). So here we have 6 unknowns in total but only 3 equations. We cannot solve for all the unknowns. We could use a shadow analyzing algorithm in which we can determine the pixels under the shadow that is caused by a particular light source we then can calculate the individual portion of the light source contribution. However, this only work if the light sources are far apart that create two easily distinct shadows.