COMP 5421 Project 4

Website

`https://github.com/Tony-Mao/shapeFromShading`

Method

1. Calibrate the lighting direction. $\vec{L} = 2\vec{N}(\vec{N} \cdot \vec{V}) - \vec{V}$, where $\vec{L}$ is the lighting direction, $\vec{V}$ is viewing direction of camera which by default is set to be viewing from top, and $\vec{N}$ is surface normal which can be calculated based on the calibration object.

2. The raw calibration result for lighting direction is not very acceptable by the limit of sampling. Hence subsampling is preferred. One can reunion close light direction (and their associated images) by picking light direction on vertices of a icosahedron.

3. In the Lambertian model, $I = \rho\vec{N} \cdot \vec{L}$, the ρ is also an unknown. In order to get rid of it, we can assign one image to be 'denominator image', so that $I_1/I_2 = \vec{N_1} \cdot \vec{L_1}/\vec{N_2} \cdot \vec{L_2}$. The denominator image should contain least shadows and highlights. One can use average intensity over whole image to get a 'best' image, or one can get 'best' intensity for each pixel and merge them to be a denominator image.

4. Solve equation $I = \vec{N} \cdot \vec{L}$ by SVD. After dividing denominator image, one will get $I_k/I_d = \vec{N} \cdot \vec{L_k}/\vec{N} \cdot \vec{L_d}$, where I_k is intensity at certain pixel of k^{th} image, I_d is intensity at certain pixel of denominator image, L_k is light direction for k^{th} image and L_d is light direction for denominator image. To solve $\vec{N}$ at each pixel, we write $\vec{N} = (x, y, z)$. and $\vec{L_i} = (L_i x, L_{iy}, L_{iz})$. Then we can get

$$(L_k x - (I_k/I_d)L_{dx})x + (L_k y - (I_k/I_d)L_{dy})y + (L_k z - (I_k/I_d)L_{dz})z = 0$$

Then we let k varies in total numbers of sampling (skip denominator image). Then we get a huge matrix M with each row as $(L_k x - (I_k/I_d)L_{dx}, L_k y - (I_k/I_d)L_{dy}, L_k z - (I_k/I_d)L_{dz})$. We

apply Singular Value Decomposition(SVD) on this matrix, and the 'V-vector' corresponding to smallest σ is the solution of surface normal.

In Matlab, things are just as simple as

```
[~,~,V] = svd(A);
sol = V(:,end)
```

5. Use MRF Graph cut to get a better result. Details refer to Tai-Pang Wu and Chi-Keung Tang, **Dense Photometric Stereo Using a Mirror Sphere and Graph Cut**, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005)

6. Reconstruct shape from Shape-From-Shapelets method. Details refer to Peter Kovesi, **Shapelets Correlated with Surface Normals Produce Surfaces**. *IEEE International Conference on Computer Vision*. (2005) pp 994-1001. Toolbox from `http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/`