Read me

Challenge 1:

findSphere:

Radius:

First transform the image into binary image, then sum up the image matrix the result is the area of circle thus to calculate radius.

Center:

Use 'find()' function to find the maximum and minimum index of columns and rows, then calculate the average of max and min as the center.

computeLightDirections:

As we made the assumption that the diffuse reflection is resulted from a single light source, thus the 'relative brightness' (which one is brighter and which one is dimmer) of a point on the surface depends only on $(\vec{n} \cdot \vec{s})$, the brightest point should have a normal $\vec{n}$ which is closest to the light source direction $\vec{s}$ among all the surface normal. Thus it is safe to assume this.

In this part, find the brightest point (if there are multiple points, take the average) and calculate the vector from circle center to this point, this vector is the direction in x-axis and y-axis. Then use the radius gained in 'findSphere' to calculate the direction in z-axis. Finally, normalize the direction vector to have length 1.

computeMask:

First initialize an all-zero mask matrix with same size of the image, then add each image to the mask matrix, finally set all non-zero value in mask matrix to 1.

computeNormals:

I used all five light sources in this part to calculate the norm. The method is the one given in lecture $N = (S^*S)^{-1}S^*I$. Then normal is $\frac{N}{\|N\|}$ and albedo is $\|N\|$

Challenge 2:

generateIndexMap:

First calculate 'similar laplacian' $\nabla_M^2 f$ of each pixel (pad the image with 0), then calculate the sum within the window (pad the matrix with 0), finally find the index of stacks which has the highest sum for each pixel.

loadFocalStack:

One thing to note, here I stored the image stack in a cell rather than a matrix for that cell is more flexible in size.

refocusApp:

First show a random image from the stack, then enter an infinite loop, in this loop, use 'ginput()' function to get a pair of coordinate, if the coordinate is out of the bound of image, exit the loop and halt the function, else get the value of the coordinate on index map, and then display the images between current index and the new index, finally, update current index with new index.

When you click on the image to choose a point, hit 'enter' to input the value, else the coordinate will not be input. I set a 0.3s pause in the change of image, otherwise there will be a sudden change and you may not see the change of images, so be patient while the image is changing.