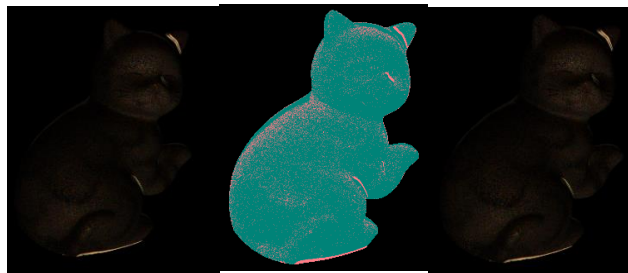


Part 1: Photometric Stereo

Date: 2019-4-5

- Estimation of normal:
 - The attempt of continuous sequential image

Firstly, I tried to process the pictures based on the 3 continuous sequential images: 030.png, 031.png, and 032.png, the result was not ideal, there was much noise in the picture, and large area of the same color, as shown below: (take the cat as an example)



In the file 'bearPNG_Normal.txt', we can see that the data of normals are so similar that they can't be ideally represent by the RGB color that range from 0 to 255.(because the mapping from floating point numbers to integers lead to the loss of information)

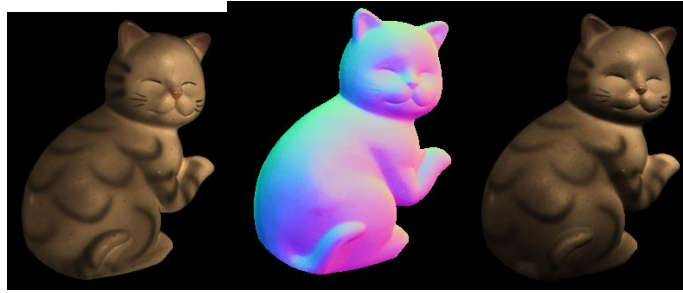
[illegible]

And from the file 'light_directions.txt', it is obvious that the direction of the 3 light are really similar, it is where the problem lies

We can regard the L^{-1} as the base of vector coordinate system, and the \mathbf{I} is \mathbf{b} 's coefficients, it is obvious that the above base is unsuitable, so that the result is not ideal

- The attempt of uncontinuous sequential image

After the simple analysis of the problem, choose the images: 030.png, 050.png, and 070.png, we can get a not bad result: (take the cat as an example)



From left to right are the surface-albedo, the normal, the re-rendered picture(the light from the direction $[0,1,0]$), and we can contrast the different results using different data as follow:

Table:

| Picture number | L | L^{-1} | intensity | reslut | |
|----------------|--|--|--|--------|--|
| 1,2,3 | $\begin{bmatrix} -0.0635 & -0.4317 & 0.8998 \\ -0.0629 & -0.3178 & 0.9461 \\ -0.0612 & -0.1901 & 0.9799 \end{bmatrix}$ | $\begin{bmatrix} 1.0e+05 * \\ -2.7933 & 5.3499 & -2.6004 \\ 0.0793 & -0.1519 & 0.0739 \\ -0.1591 & 0.3047 & -0.1481 \end{bmatrix}$ | $\begin{bmatrix} 1.3 & 1.5873 & 2.1503 \\ 1.4726 & 1.8014 & 2.4831 \\ 1.5606 & 1.9232 & 2.7339 \end{bmatrix}$ | | The L^{-1} is too big |
| 30,50,70 | $\begin{bmatrix} -0.4162 & 0.2089 & 0.885 \\ 0.0478 & -0.3228 & 0.9453 \\ 0.2954 & 0.2046 & 0.9332 \end{bmatrix}$ | $\begin{bmatrix} -1.4217 & -0.0399 & 1.3886 \\ 0.6744 & -1.8677 & 1.2524 \\ 0.3022 & 0.4221 & 0.3574 \end{bmatrix}$ | $\begin{bmatrix} 0.8686 & 1.064 \\ 1.4264 \\ 0.7568 & 0.9313 & 1.2723 \\ 0.6401 & 0.7827 & 1.0607 \end{bmatrix}$ | | Not bad |
| 1,44,87 | $\begin{bmatrix} -0.0635 & -0.4317 & 0.8998 \\ -0.614 & -0.0284 & 0.7888 \\ 0.4875 & 0.2996 & 0.8201 \end{bmatrix}$ | $\begin{bmatrix} -1.3335 & 0.0163 & 1.6119 \\ -3.4833 & 2.1782 & 1.9200 \\ -0.6539 & 1.0462 & 1.0349 \end{bmatrix}$ | $\begin{bmatrix} 1.3 & 1.5873 & 2.1503 \\ 0.561 & 0.6826 & 0.8794 \\ 0.306 & 0.3693 & 0.476 \end{bmatrix}$ | | The derrection of L^{-1} is not good |

And we can also detect another problem: the shadows(left) and highlights(right)

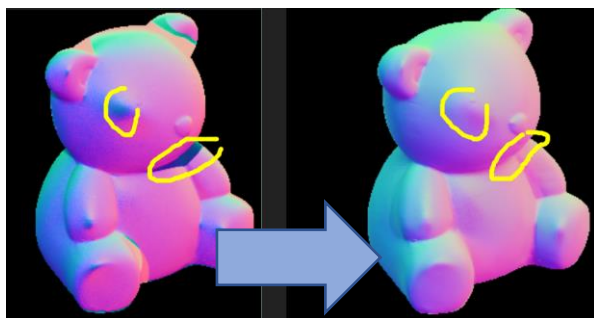


And in the next section, a sort operation can solve these problems.













■ Final optimization

To optimize the result we simply get from 3 images,use the sort operation to solve the problem that caused

by the shadows and highlights, we get the results of 35 groups of images(each contains 3 images), and then for each pixel and each direction of normal, sort them and discard the data that is too large or too small, and finally we can get the ideal results.



■ Final result: (use the illumination direction of $[0,0,1]$)

| | Normal | Albedo map | re-rendered picture |
|--------|---|--|---|
| buddha |  |  |  |
| pot |  |  |  |
| cat |  |  |  |
| bear |  |  |  |

■ Conclusion

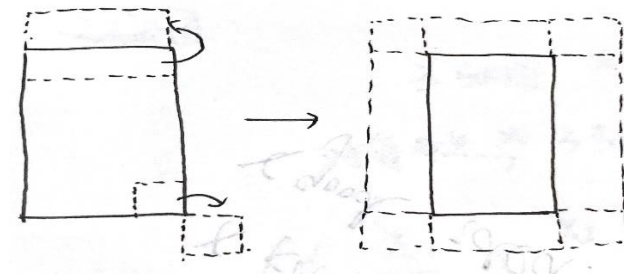
In order to get a ideal result , the insensity and the direction of lights should have relatively large difference, otherwise, the normal can not be presented visibly.

Part 2: Image Filtering

■ Convolution

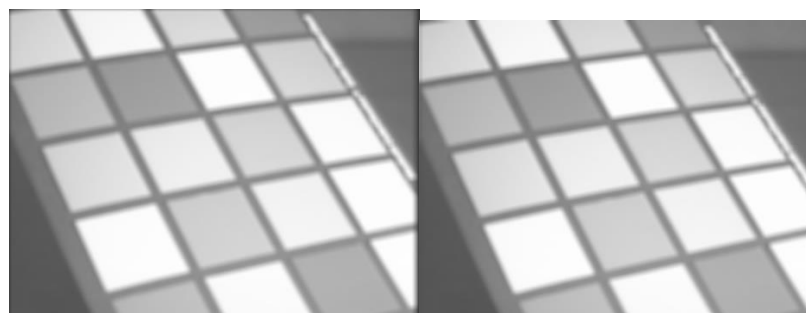
In this section, in order to handle boundary cases on the edges of the image , it is imperative to pad the image such that pixels lying outside the image boundary have the same intensity value as the nearest pixel that lies inside the image.

A convenient way to do that is to turn over certain part of the image:



And then use the matrix h to point multiplication with the local window and then get the convolution, the following are the results of Convolution when h is ones (10,10) and gaussian kernel;

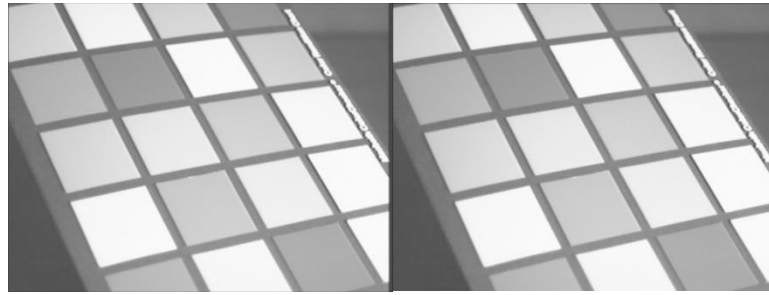
$h = \text{ones } (10,10)$



My Convolution

Sample

$h = [1,2,1 ; 2,4,2 ; 1,2,1]$ (gaussian kernel)

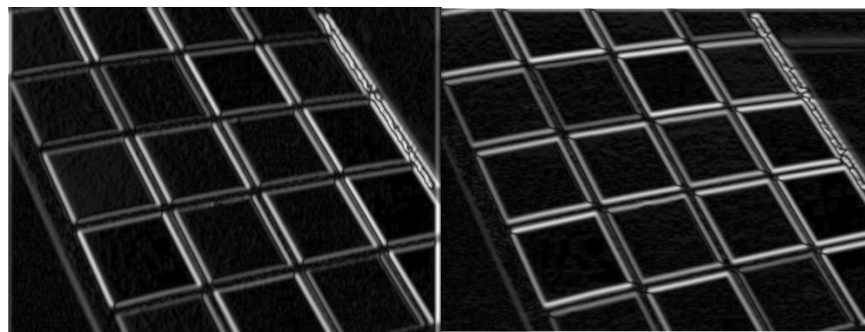


My Convolution

Sample

■ Edge detection

First, the sobel filter should be used to abstract the edge that in the x direction and the y direction, the process is similar to the last section.

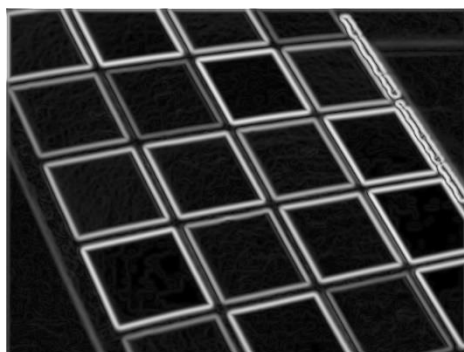


x direction

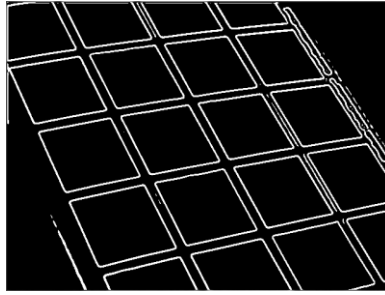
y direction

Then, a image that combine the two images is required. Suppose the result of the two direction's sobel filter is G_x, G_y , and in order to get the gradient:

$$G = \sqrt{G_x^2 + G_y^2}$$



This image has much noise and then we calculate the global average and for each window, calculate the local maximum and minimum gray value, those pixels whose gray value smaller than the global average (too dim) or $(\max - \min)$ is too small (the smooth area) will be directly set to 0.



The result looks not bad, and then it is obvious that the edge is too thick. And then non-maximum suppression can be used to solve the problem. My Non-maximum suppression is divided into two processes: the x direction and the y direction. Take the x direction as an example, if we observe a edge pixel that the difference between its upper and lower sides is relatively large, it means the edge is in the x direction, we need to find the maximum value of a continuous sequence of pixel and set it to 1, the others set to 0.

