Implement:

In Photometric Stereo, we assume that if we fixed the camera and the object, only change the light source position, and take the few photos then we can reconstruct the object in 3D by the few 2D image. the idea is that first we need to find the normal map, normal can be calculus by the following equation:

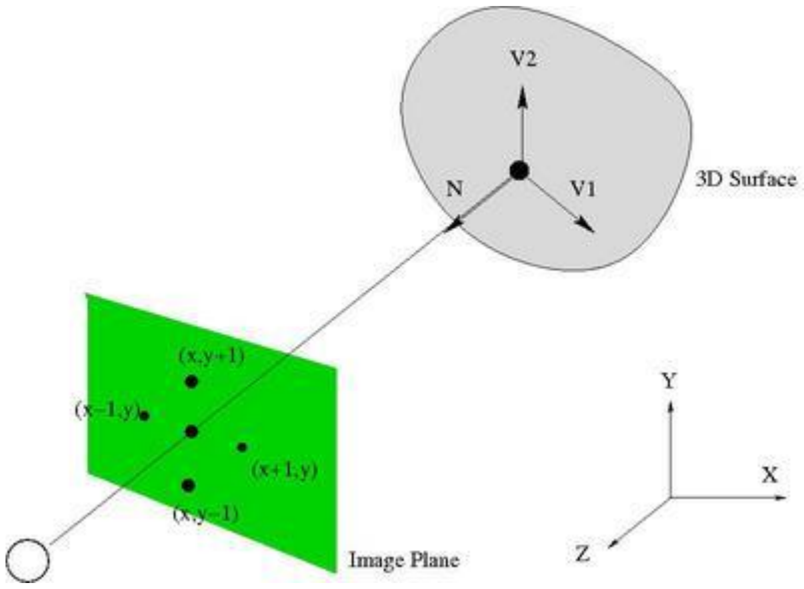
$$I = LN$$

I is our photo information, L is the reflected light direction, N us the normal at the surface point, we need at least three light source which are not in the same plane. In this equation we can solved this N by multiplied pseudoinverse of L, like the following equation.

$$L^{+}I = L^{+}LN$$

$$L^{+}I = N$$

With the normal map then we can do the surface reconstruction by compute the depth map, each pixel in the normal map N must be orthogonal to the vector V1 and V2.



$$\begin{aligned}
V_1 &= (x+1, y, z_{x+1,y} - (x, y, z_{x,y}) \\
&= (1, 0, z_{x+1,y} - z_{x,y}) \\
N.V_1 &= 0 \\
(n_x, n_y, n_z).(1, 0, z_{x+1,y} - z_{x,y}) &= 0 \\
n_x + n_z(z_{x+1,y} - z_{x,y}) &= 0
\end{aligned}$$

$$\begin{aligned} V_2 &= (x, y+1, z_{x,y+1} - (x, y, z_{x,y}) \\ &= (0, 1, z_{x,y+1} - z_{x,y}) \\ N.V_2 &= 0 \\ (n_x, n_y, n_z).(0, 1, z_{x,y+1} - z_{x,y}) &= 0 \\ n_y + n_z(z_{x,y+1} - z_{x,y}) &= 0 \end{aligned}$$

$$\begin{aligned} -n_x + n_z(z_{x-1,y} - z_{x,y}) &= 0 \\ -n_y + n_z(z_{x,y-1} - z_{x,y}) &= 0 \end{aligned}$$

Those relation can be writed as the linear system, and like the same skill, we can solve this by the pseudoinverse. z is our depth map.

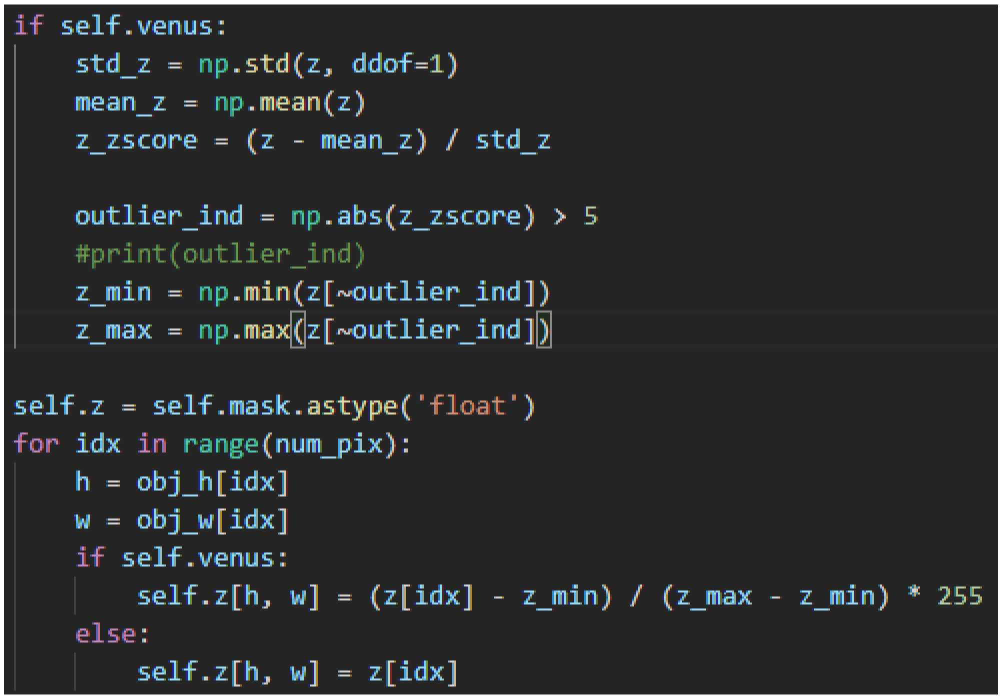
$$Mz = v$$

$$M^{+}Mz = M^{+}v$$

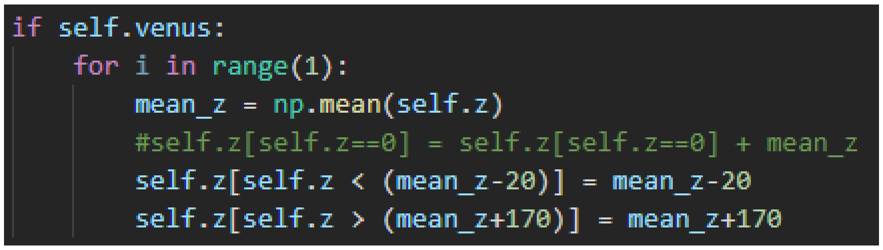
$$z = M^{+}v$$

What kind of method you use to enhance the result?

In the bunny and star case, basic method can already have well performance, but in the Venus case the body can't display well at the depth map, so we must add more effort to enhance the display result, for example first we need to compute standard deviation, mean and the z-score, after computed we can also get the min and max value, final distribute the value in the range 0 ~ 255. After normalizing we can notice that most of the pixel are far away to the min value pixel, so we do some adjusted by limited the range of the value. Implement will post at the following screenshot.



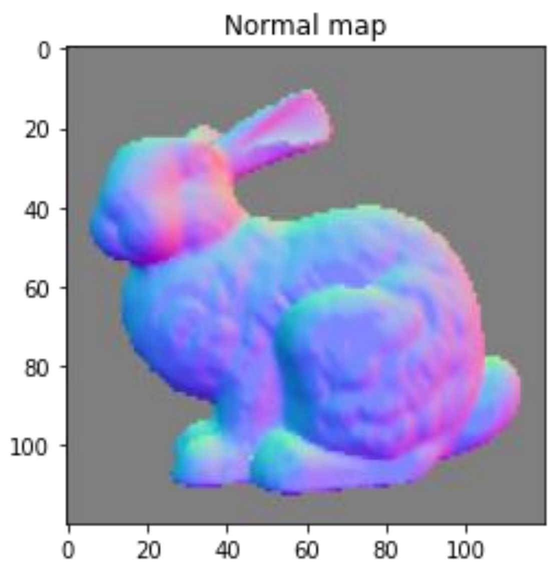
```
if self.venus:
    std_z = np.std(z, ddof=1)
    mean_z = np.mean(z)
    z_zscore = (z - mean_z) / std_z

    outlier_ind = np.abs(z_zscore) > 5
    #print(outlier_ind)
    z_min = np.min(z[~outlier_ind])
    z_max = np.max(z[~outlier_ind])

self.z = self.mask.astype('float')
for idx in range(num_pix):
    h = obj_h[idx]
    w = obj_w[idx]
    if self.venus:
        self.z[h, w] = (z[idx] - z_min) / (z_max - z_min) * 255
    else:
        self.z[h, w] = z[idx]
```

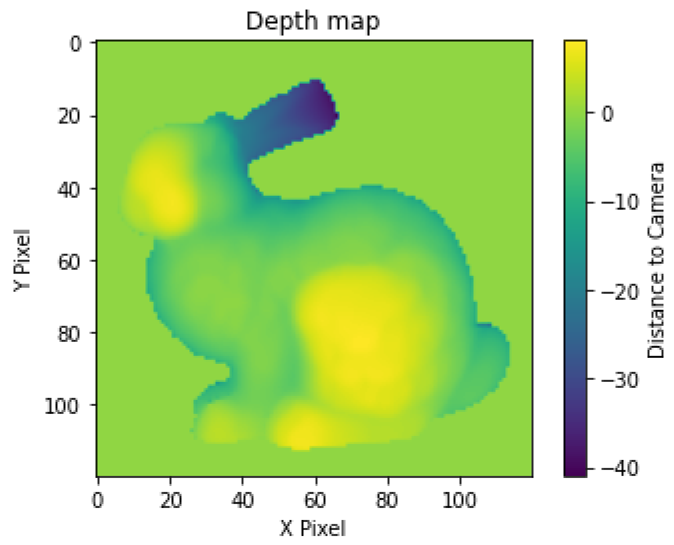


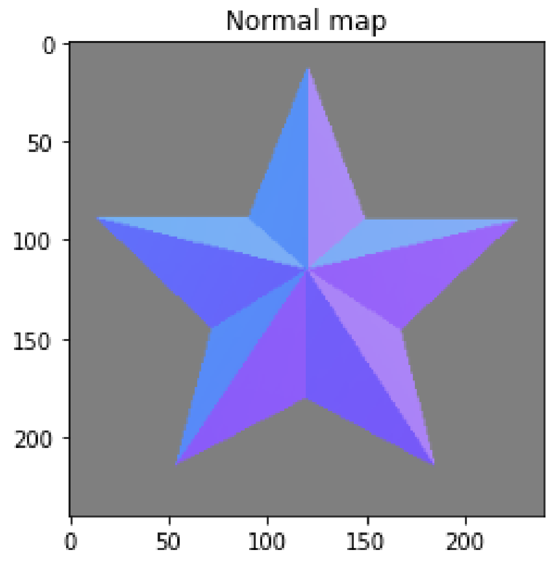
```
if self.venus:
    for i in range(1):
        mean_z = np.mean(self.z)
        #self.z[self.z==0] = self.z[self.z==0] + mean_z
        self.z[self.z < (mean_z-20)] = mean_z-20
        self.z[self.z > (mean_z+170)] = mean_z+170
```

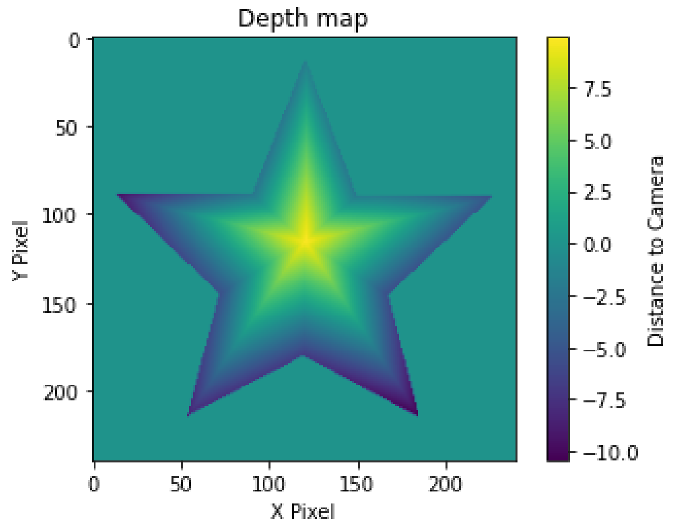
Compare the result:

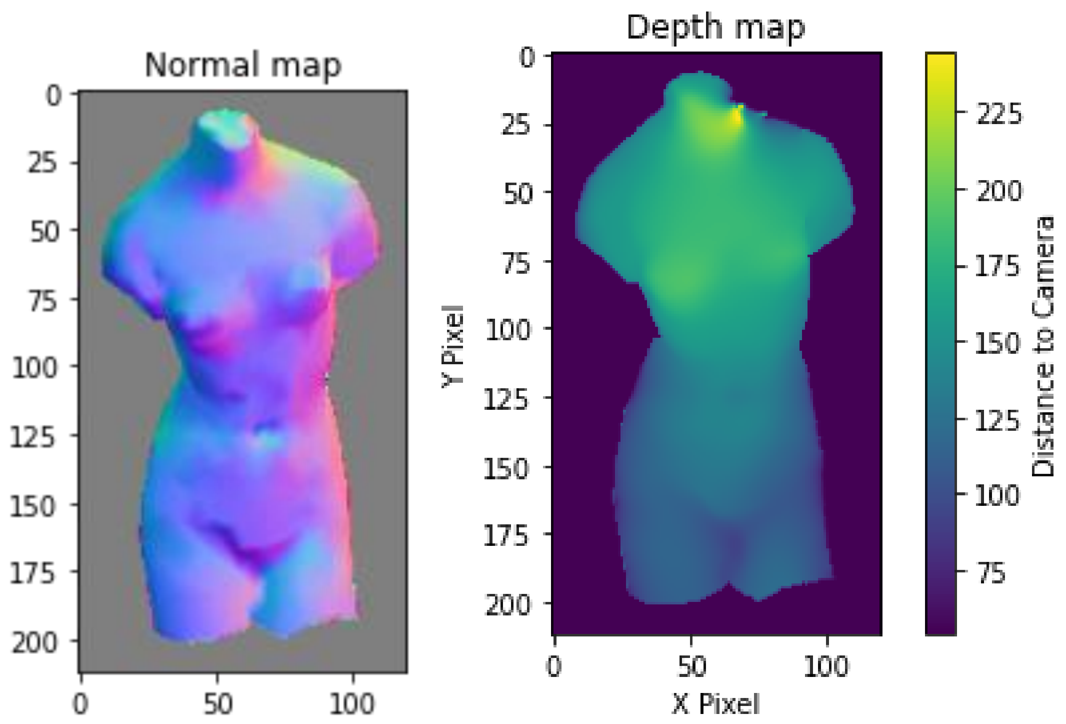
In the following I posted the normal map, depth map, and the .ply file. The most interesting part is the venus case, we can see the obvious different between adjust, before the adjust we can see that the most of the body point are far away with the base, and we can't see the detail and the depth information at the body pixel, and after the adjust we can see different color to the pixel, whole picture show the post of the venus with the clearly depth detail.


Normal map
0
20
40
60
80
100
0
20
40
60
80
100


Depth map
Y Pixel
X Pixel
Distance to Camera
0
20
40
60
80
100
0
20
40
60
80
100
0
−10
−20
−30
−40


Normal map
0
50
100
150
200
0
50
100
150
200


Depth map
Y Pixel
X Pixel
Distance to Camera
0
50
100
150
200
0
50
100
150
200
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0


Normal map
Depth map
Y Pixel
X Pixel
Distance to Camera