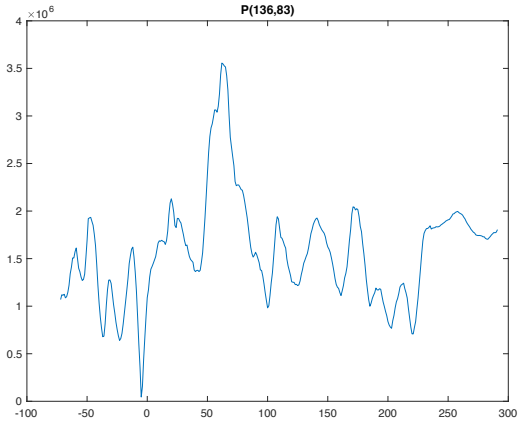
Computer Vision HW5

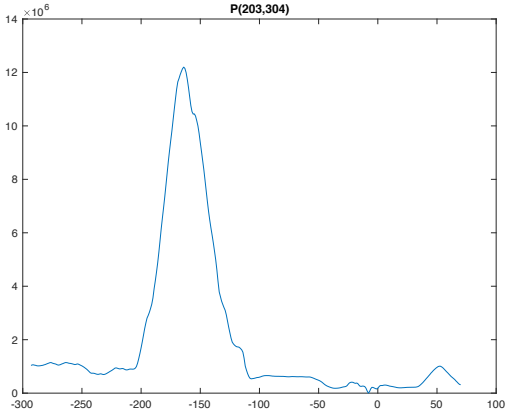
(Aaron) Chien-Cheng Lai
ChBE

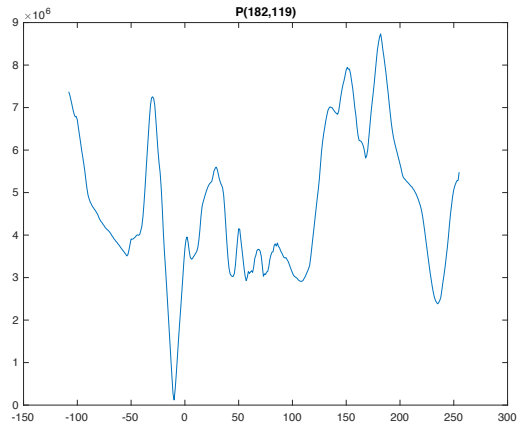
Solving Correspondence

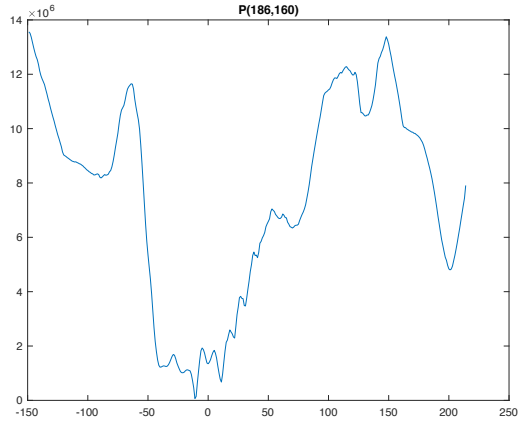
1. I choose 21 for patch size and pixel-wise sum of squared distances(SSD) for similarity metric. The lager patch size would cause more computational cost but reduces more noise in the disparity map. After several tests, I choose 21 for patch size because of this trade-off. Also, I think SSD would be a good choice due to low computational cost and good result.

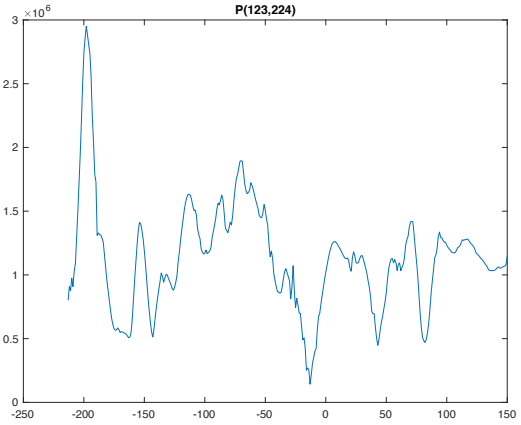
2. The 6 similarity profiles for each pixel shows below.

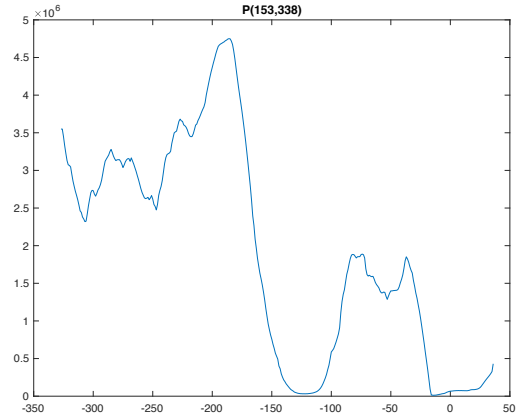






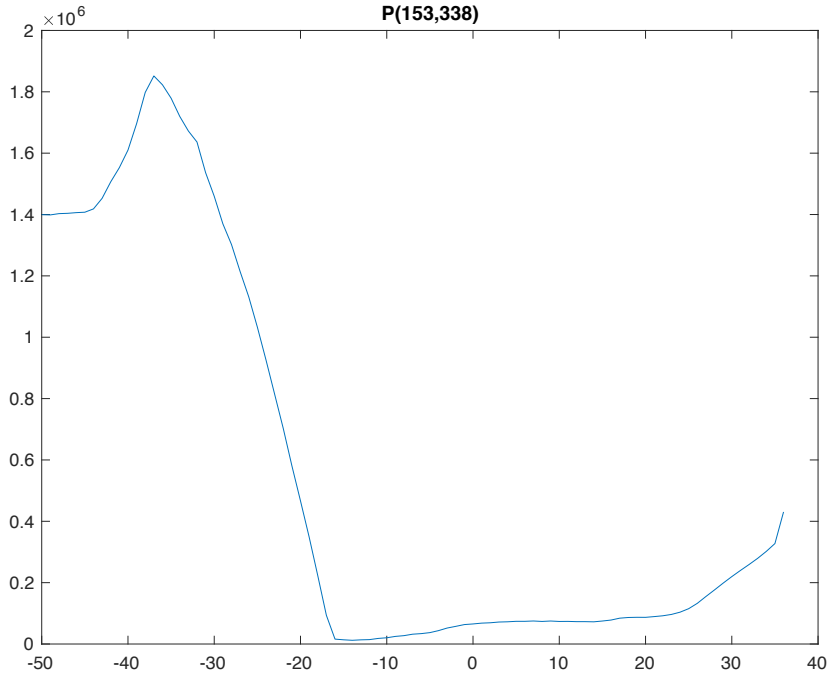






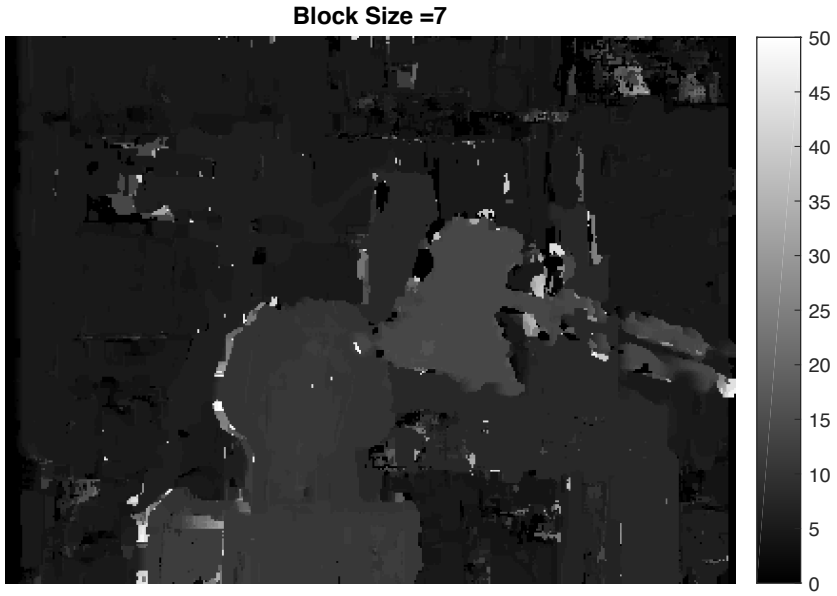
From those similarity profiles, I think it is reliable since we can find the obvious minimum value in each figure and the disparity of each figure also has similar value. Although it seems to have two minimum values in p6(153, 338) figure, we can still find the wanted disparity in the right one due to its expected disparity.

3. In order to find the best match of correspondence for each pixel, I sort the value of sum of squared distances and look for the smallest one. Besides, the disparity of each pixels is supposed to be in a reasonable range. That is to say, we should find the corresponding pixel whose coordinate is near the original one. Therefore, we can set a range for disparity to find the reasonable disparity. I set 50 for range size and the p6(153, 338) figure shows expected result as below.



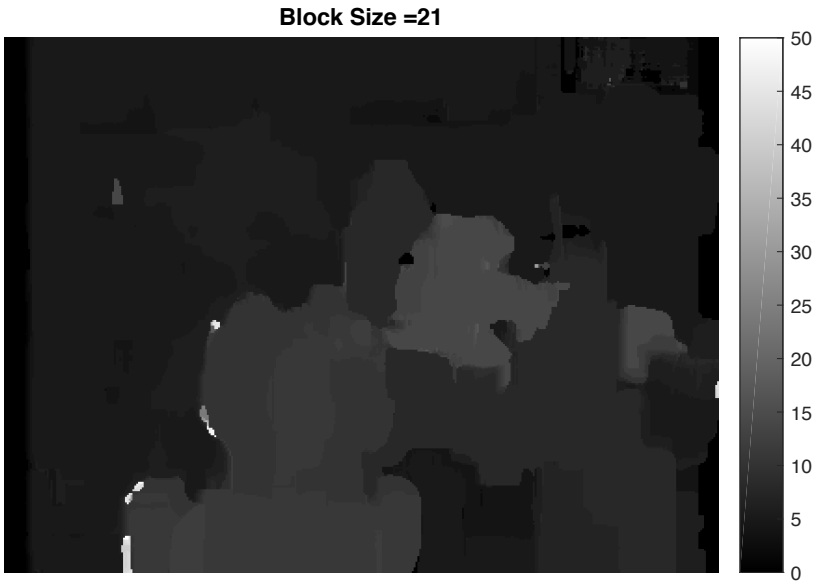
We can find the disparity is between -10 and -20 which is the same as we expect previously so I think a reasonable range helps us find disparity we want.

4. By setting 7,11,15 and 21 for bock size.



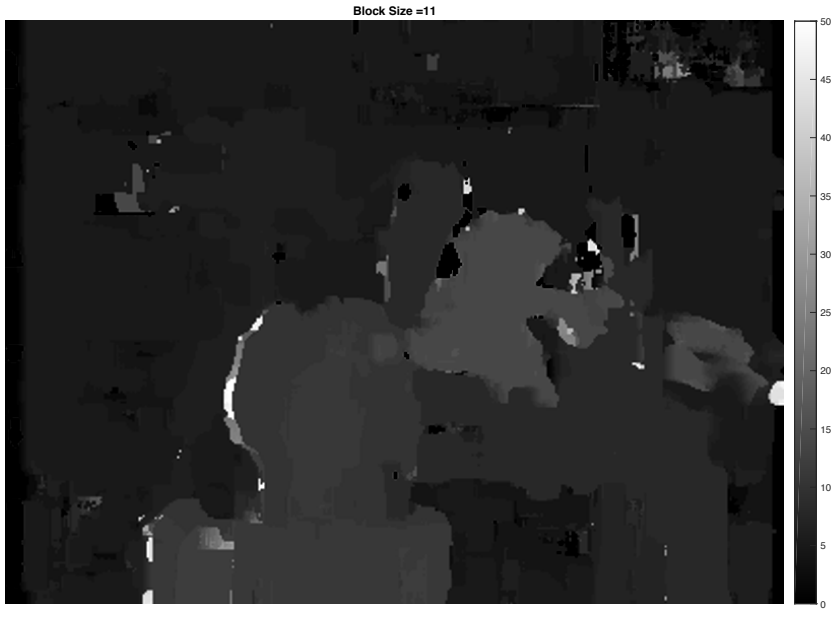
Block Size =11

Block Size =15



As the block size increases, the disparity map becomes smoother but more computational cost as well. The undefined area is on the rows between [1, half patch size] and rows between [total columns - half patch size, total columns].

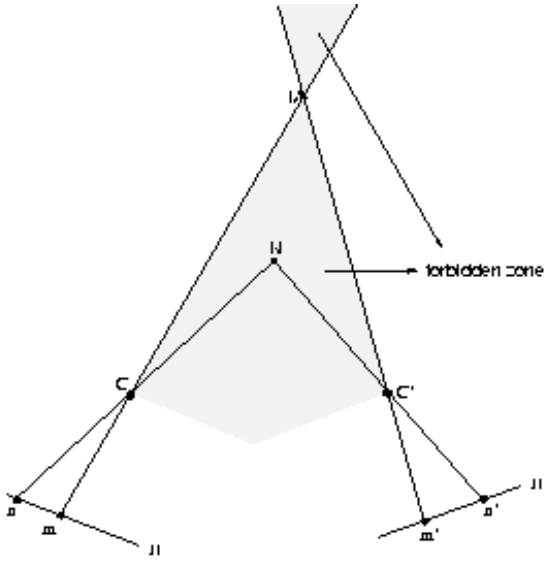
5. I choose cubic for interpolation which increases the pixels of image.



As result shows, it indeed reduces some noise but doesn't have much influence. However, I don't have unknown disparity in the map because I've add the boundary condition in my code.

6. In order to deal with the stereo matching problem, we have to apply some constraints, including smoothness, uniqueness and ordering.

    - Smoothness: We expect that disparity of pixels should change from each other smoothly along the whole image. However, if there is a discontinuity depth area, it would fail and result in drastic change in disparity.
    - Uniqueness: We expect a given pixel should not have more than one matches to the other image. However, it fails when there is an uniform or repetitive area in the image.
    - Ordering: Consider a case that two pixels A and B match the pixels A' and B'. If A is on the left of A', B should be on the left of B' too. However, it fails at the following region. The orders of n and m are inverse in two images.

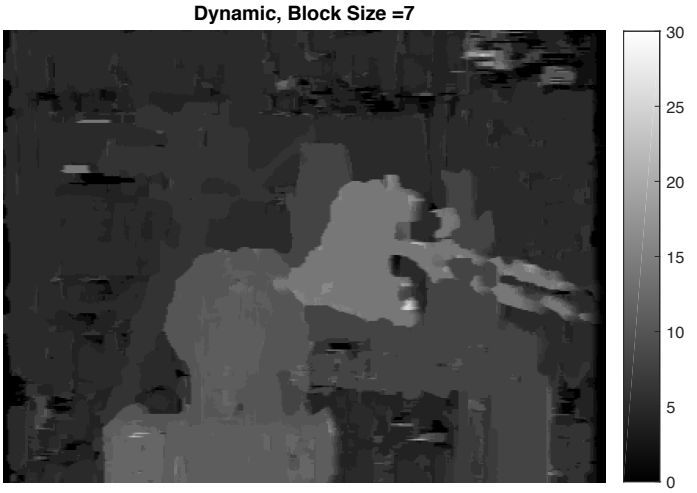


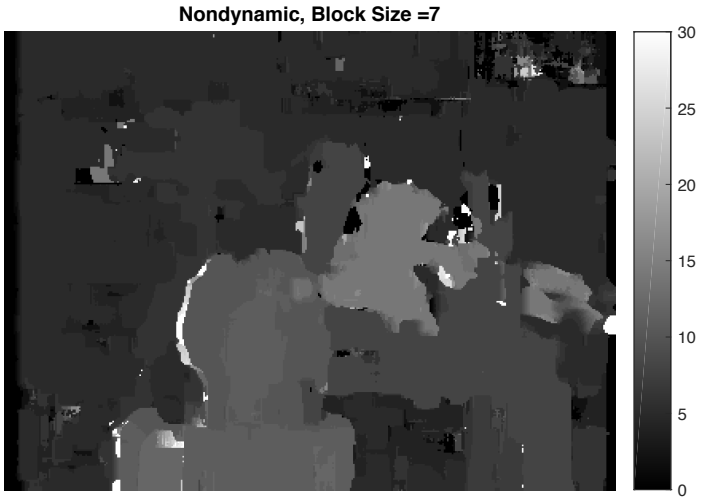
Scanline stereo

1. This paper is mainly talking about how we can find the disparity between two images by implement of intra and inter scanline. Moreover, because it would cost a lot computational resource when we do the intra and inter scanline, so we can apply dynamic programming to search both simultaneously which would reduce lots of the computational cost.

   Therefore, due to the benefits of smoothness, uniqueness and ordering constraints we discussed previously, we could improve our stereo matching result by applying them to dynamic programming.

2. By setting 7 for patch size and 30 for disparity. I run the dynamic programming and result shows below.





We can found that dynamic programming really improves the performance in disparity map. It reduces a lot noises and allow us to see the contour more clearly.

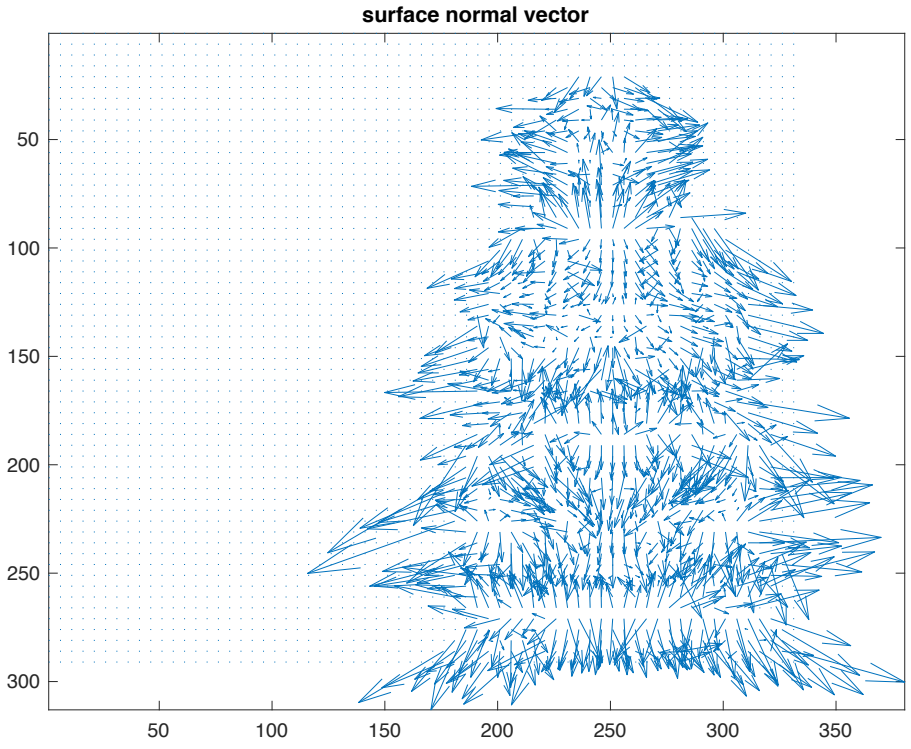
Photometric Stereo for Lambertian object

1. Firstly, I separate all images into three parts, red, green amd blue and do the programming separately. For each color channels,

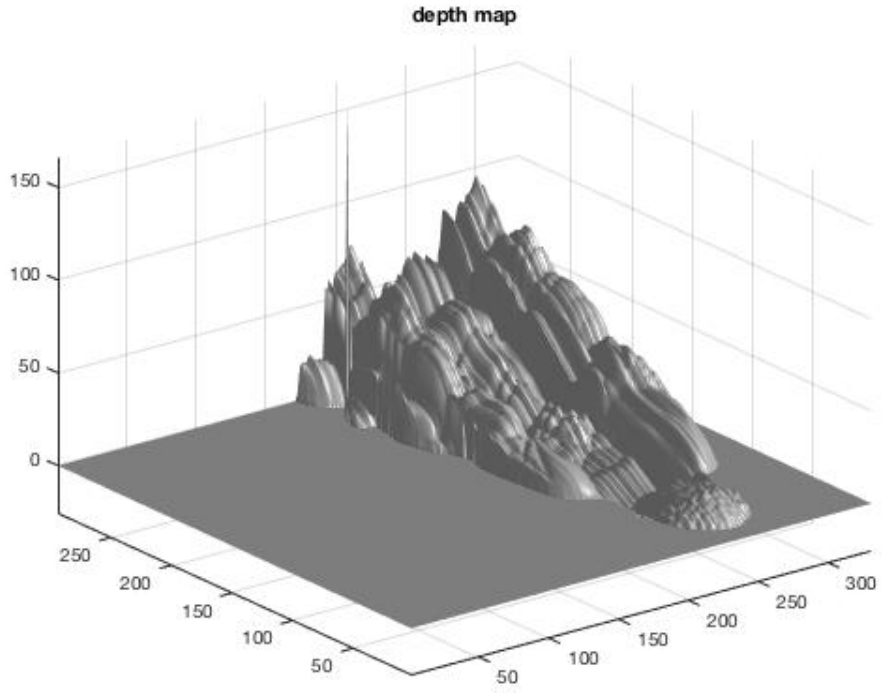
red

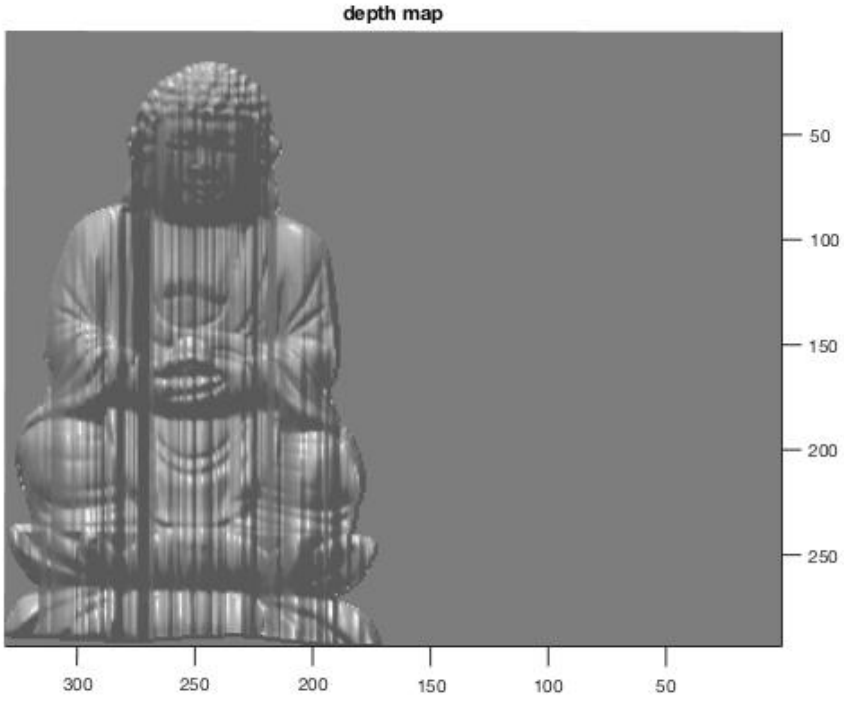
red

2. Use quiver(1:5:ncols,1:5:nrows,p(1:5:end,1:5:end),q(1:5:end,1:5:end),10) to compute the surface normal vector.



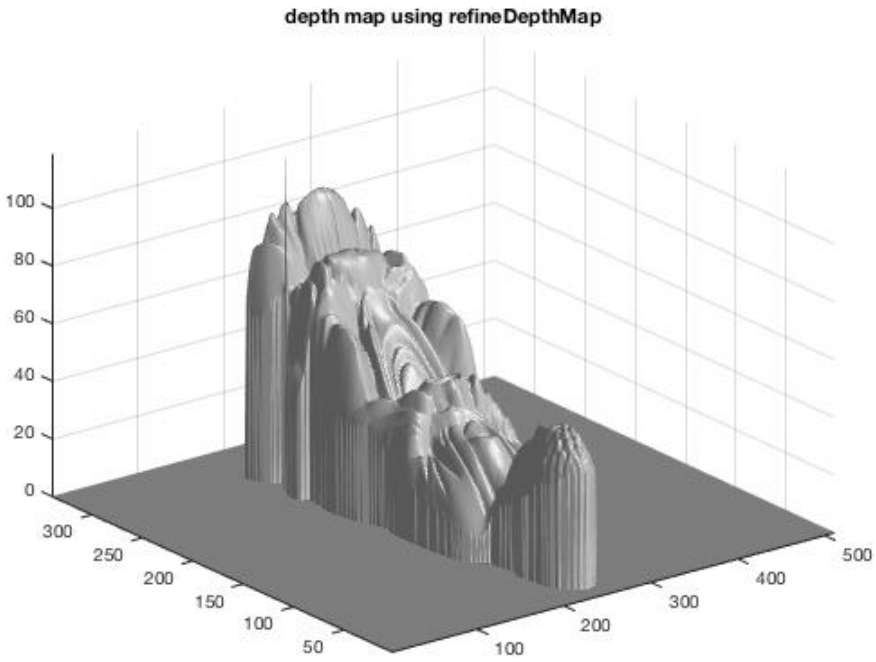
3. Use the computed surface normal vectors to find a depth map by simple integration and using the provided p-code.





This depth map still has some inperfect points and area. That is, the code still need to be improved to make result better.

4. Use refineDepthMap function to generate depth map.



Comparing those two different way in depth map, the basic intergration is definitely not good enough to perform the map but allow me to understand the fundamental algorithm that computes the depth map.