ECE 6310- INTRODUCTION TO COMPUTER VISION

LAB 8: RANGE IMAGE SEGMENTATION

Submitted To:

Dr. Adam Hoover

Ashit Mohanty

C13582787

amohant@g.clemson.edu

Date of submission: 12-03-2020

INTRODUCTION

The main objective of this lab was to segment a range image based upon surface normal. A range image of the chair was provided, chair-range.ppm, which was the main image to work upon. The output of this lab is three-fold. Firstly, the original image is masked based on the pixel values. Secondly, the x, y, and z coordinates for every pixel are computed by the formula provided. These x, y, and z coordinates are then used to compute the surface normal for every pixel, that falls in the threshold category as mentioned in step 1. These surface normals are then stored as an output in a text file. These surface-normals were better visualized using a quiver plot. The third output is the segmented image, with each separate color specifying pixels having a similar surface normal. This way the pixels that are assumed to be of one surface have the same color.

This range image segmentation program was written in C and the algorithm for segmentation was performed on the original image, chair-range.ppm, as attached underneath in Figure 1.

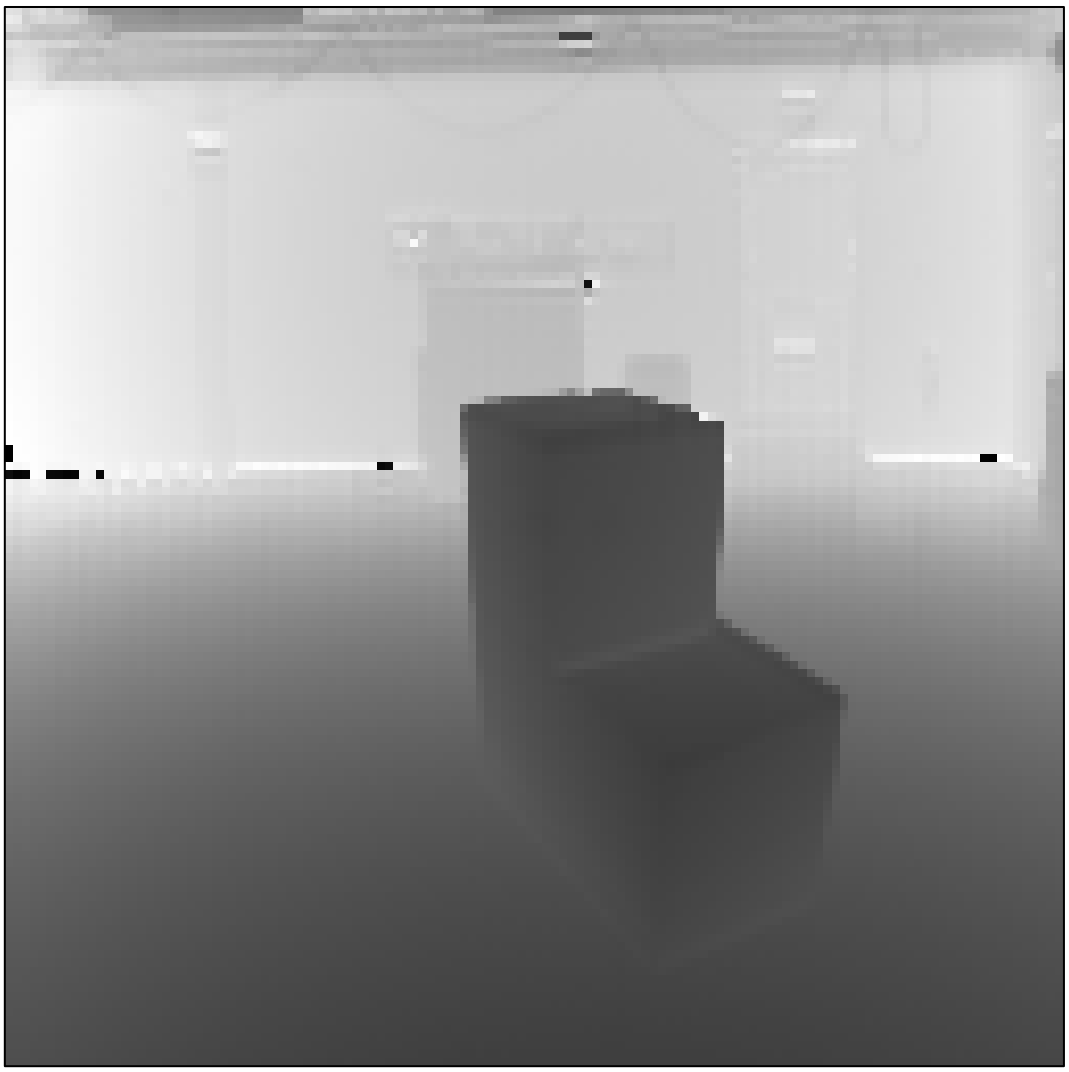

Figure 1: Original image of chair-range.ppm

MASKING ORIGINAL IMAGE

First, the original greyscale image is masked at a particular threshold value to convert the image to a binary image. The pixels that fall below the threshold value are assigned a value of 0, and the values that are above the threshold are assigned a value of 255. The threshold value is chosen such that only the chair and floor get the value of 0. **The threshold value that was selected for creating the binary image was 140.** So, any pixel value that was lesser than or equal to 140 was assigned the value of 0. As a result, the binary image that was generated was stored in a file titled, 'chair_threshold.ppm', as shown underneath in Figure 2.

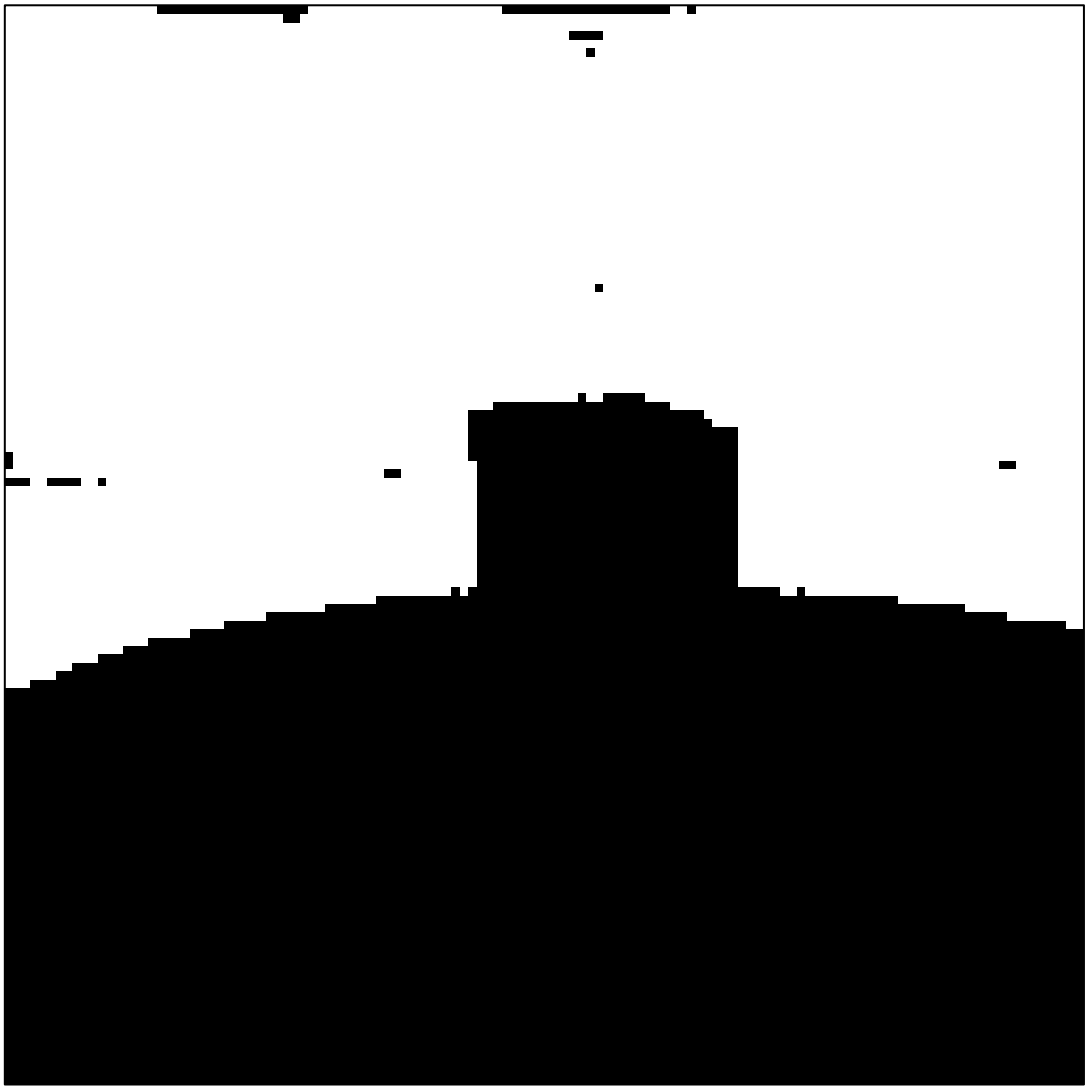

Figure 2: Binary masked image at the threshold value of 140. Stored as chair_threshold.ppm.

SURFACE NORMALS OF THE PIXELS

The next step is to compute the surface normals of all the pixels that have been masked by the previous step. The pixels that didn't lie in the threshold, i.e. had the value of 255 after masking, were assigned the normal values of 0 in the x, y, and z-direction. Firstly, the x, y, and z coordinates of each pixel were computed using the formula provided in the given C-code, 'odetics-to-coords.c'. These coordinates are stored in an output text file, 'chair_coordinates.txt'. Then these values were used to compute the surface normals using the formula for a cross product between two vectors. **The distance for the computation of the vectors was 4 units.** So two vectors were formed, one between the original pixel point and the pixel point 4 units away in the negative x-direction, and the second vector between the original pixel point and the pixel point 4 units away in the negative y-direction. The cross product was then computed by the formula,

$$C_x = A_yB_z - A_zB_y$$

$$C_y = A_zB_x - A_xB_z$$

$$C_z = A_xB_y - A_yB_x$$

Where, the form A_x indicates the x-coordinate of vector A. Hence, the output is vector C, which is the normal vector at each surface point. To better visualize the surface normals, the normal vectors were plot using the quiver3 function on Matlab indicating the direction of the vectors.

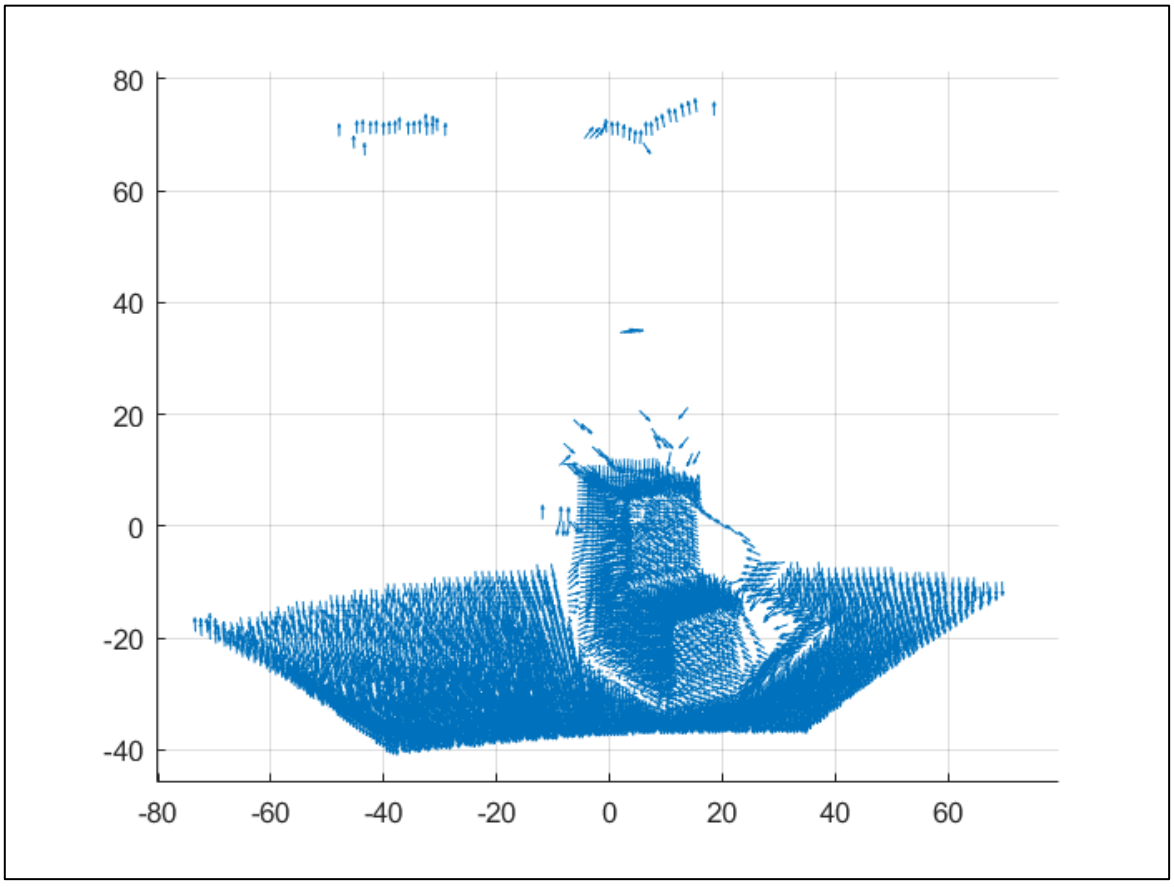

Figure 3: Quiver plot image of the normal vectors for each of the pixels.

REGION SEGMENTATION

To segment the regions, the region growing code, that was provided early on in the semester, was used as a reference. The seed pixels for the region growing was selected based on two conditions. If a particular pixel was not masked out as explained in step 1 and if any pixel in a 5x5 window around the particular pixel is not colored previously, then the pixel is considered eligible for seeding a new region. This pixel coordinate is then used as an input for the RegionGrow() function, which enables the region to grow based on if the pixels satisfy a predicate value. The logic behind a pixel joining the region is the predicate that the angular difference between the current vector and the average orientation of pixels already in the region is lesser than a particular threshold. The angle between the normal vector of the pixel that is being checked and the average of pixels already in the region is computed using the dot product formula.

$$\theta = \cos^{-1}\left(\frac{A.B}{|A||B|}\right) = \cos^{-1}\left(\frac{A_xB_x + A_yB_y + A_zB_z}{\left(\sqrt[2]{A_x^2 + A_y^2 + A_z^2}\right)\left(\sqrt[2]{B_x^2 + B_y^2 + B_z^2}\right)}\right)$$

Where, the form A_x indicates the x-coordinate of vector A

If this angle is lesser than or equal to 60 degrees, then the pixel is a part of the region. **The threshold for checking whether a pixel could be a part of the region was 60 degrees.** Hence if the computed angle, using the above-mentioned formula, was less than 60 degrees, then it became a part of the region. This is computed till no more pixels can be segmented into regions. Once a particular region has been segmented, the number of pixels colored is returned from the function, for that region.

The output of this program is a segmented image, with different colors indicating the different surfaces as detected by the range image segmentation algorithm. The output is a ppm file, titled ‘chair-color.ppm’. This is shown in Figure 4, as attached underneath.

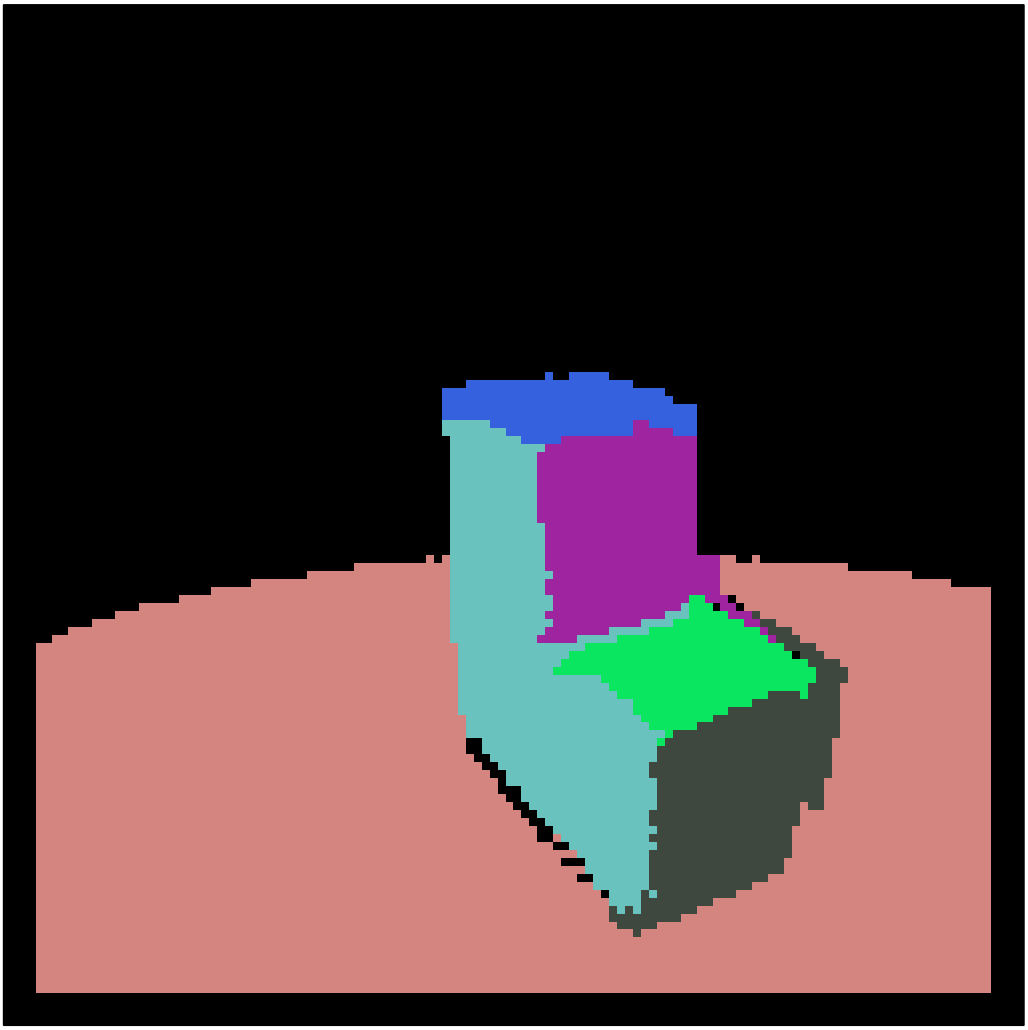

Figure 4: Segmentation output of the algorithm is stored in 'chair-color.ppm'

The image, as shown above, is further marked, along with the table showing more information regarding the average normal vector, in each direction, for each surface along with the number of pixels associated with each color region.

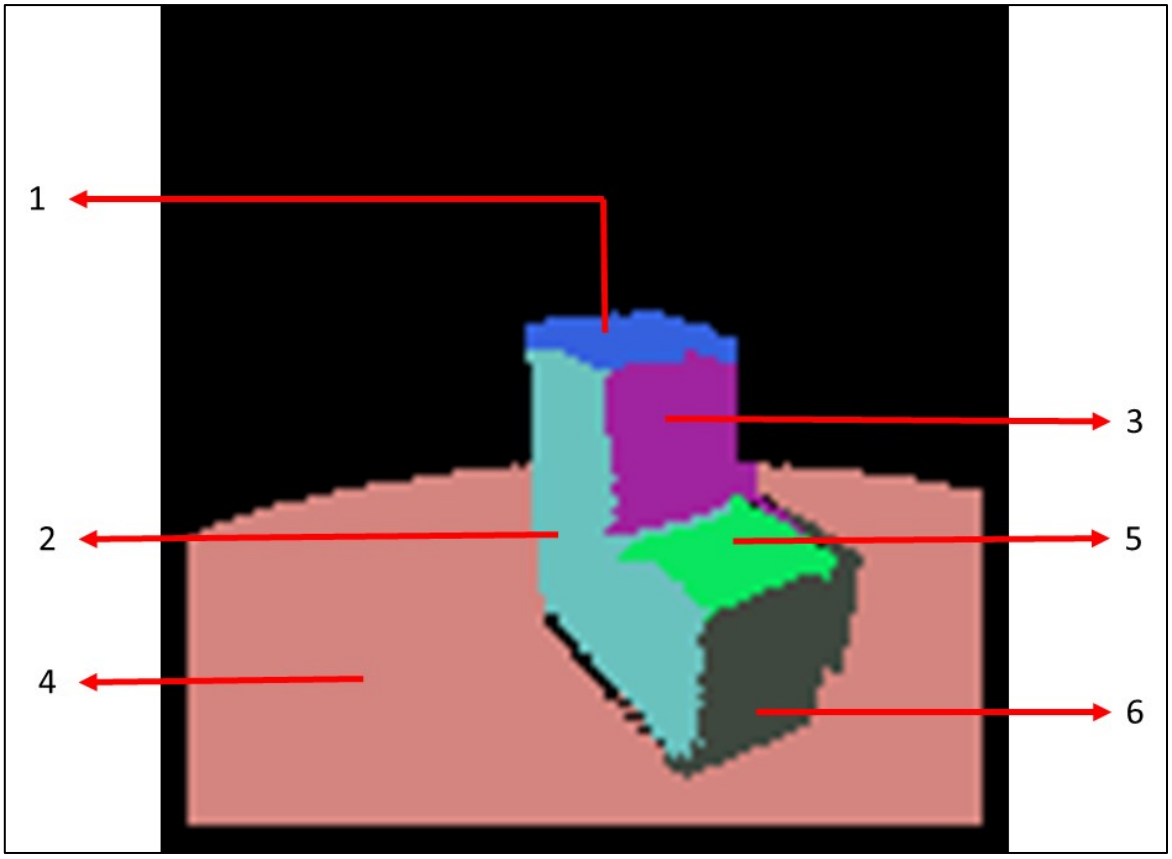

Figure 5: Marked Segmentation output of the algorithm

Region Number	Color	Number of Pixels	Average surface normal (X- direction)	Average surface normal (Y- direction)	Average surface normal (Z- direction)
1	Blue	197	9.403	-317.540	62.242
2	Cyan	845	280.502	8.321	31.732
3	Purple	493	-4.552	2.465	8.095
4	Peach	4604	3.010	-76.071	21.632
5	Lime	286	-1.343	-12.187	6.221
6	Bottle Green	529	-4.135	-4.208	8.282

Hence, the algorithm was successful in detecting each surface of the chair object using the given greyscale image. The above table indicates the description of each of the regions segmented, and Figure 4 and Figure 5 show the segmentation of the respective surfaces in the chair image, as shown in the final output file, 'chair-color.ppm'. The thresholds that were used in the program are again mentioned underneath,

- **Threshold for removing background from image (Masking): 140**
- **Distance chosen for cross-product: 4 units**
- **Angle chosen as the predicate for region growing: 60 degrees**