Selection Assignment Report

Pooya Kabiri

Visual Computing Group (VCG)

CYENS Center of Excellence

May 2021

Contents

List of Figures

1 Watershed Segmentation

1.1 Threshold

I applied softmax on the boundary probabilities, resulting in a better uniform distribution of the data.

Secondly, I used adaptive thresholding to determine the best possible threshold value.

The adaptive thresholding formula:

$$T = 2 * \frac{\sum_i^N P_i}{N} = 2 * Mean(P)$$

1.2 F1-Score

After applying the above procedure, I calculated the F1-Score using scikit-learn.

My code resulted in an F1-Score value of **98.63**.

1.3 K for KNN

I determined the best value of K empirically. I started with 3, which resulted in many small segments. My code produced the best results with **K = 5**.

1.4 Segments

By using K = 5, my code segments the point cloud into 6 segments, including **4 major segments**:

- Each two bases of the cylinder.
- Side area of the cylinder.
- Boundary points.

The two other segments are very small segments containing 1-2 points each.

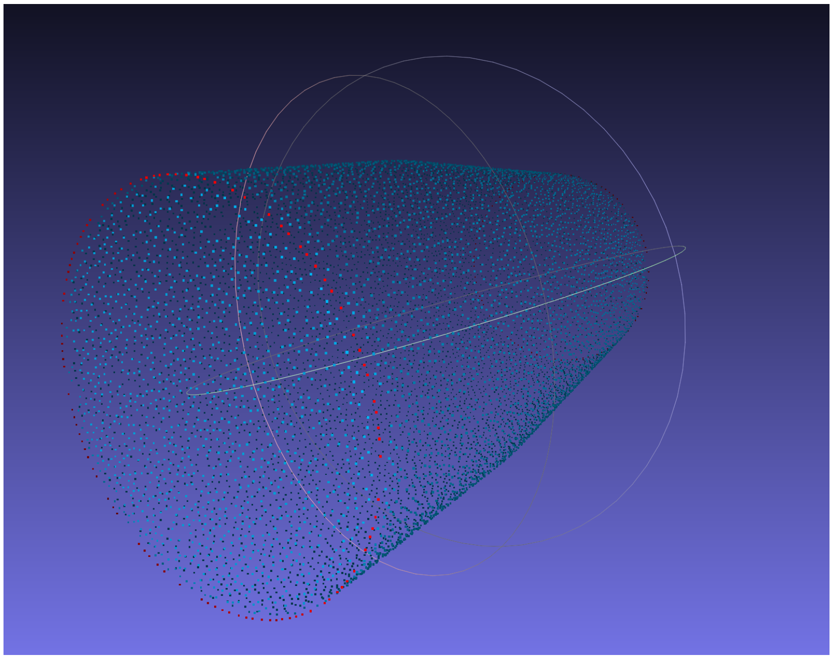

Figure 1: Threshold Boundaries - Point Cloud

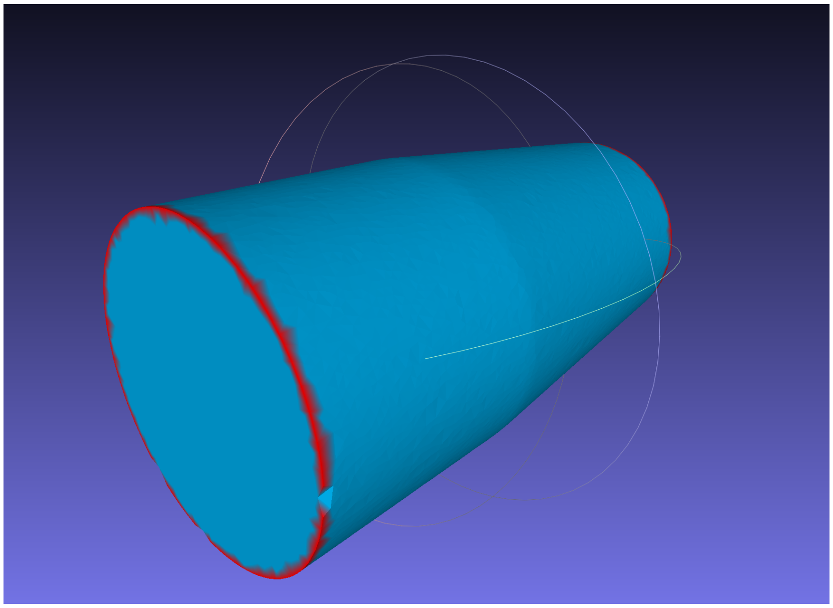

Figure 2: Threshold Boundaries - Mesh

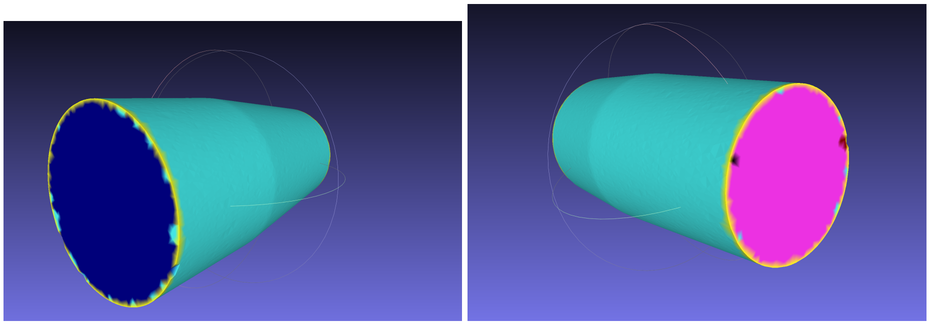

Figure 3: Watershed Segmented Mesh

Note: You can view the result of my code in `Results` folder beside the `Q1.py` file. If you wish to reproduce my results, just execute the `Q1.py` file. the scripts saves results from last run into `Results` folder.

2 Normalized Cuts Segmentation

For this section, I used `SpectralClustering` class from `scikit-learn`, with number of clusters set to 2, in this way the Spectral Clustering algorithm is actually performing normalized cuts.

I just could use this approach to segment the point cloud into two parts. I didn't have the enough time to go further on.

I found your 2020 paper on **Learning Part Boundaries from 3D Point Clouds** inspiring for this task, but unfortunately I couldn't explore your approach completely.

Link to paper website.

2.1 K for KNN

When using K values smaller than 4, the resulting KNN graph is not connected, and the clustering algorithm cannot find the appropriate segments.

I used a K value of **5**.

2.2 Best of Two Options

I segmented the point cloud into two parts, and there was no significant difference between the two approaches.

2.3 Functions

- **Angle**: I computed the normal angle using vector dot product (`numpy.dot()`) and vector norm (`numpy.linalg.norm()`).

I implemented the function `compute_angle(v1, v2)` myself. by using the abovve functions.

- **Probability**: I used `numpy.maximum()`

2.4 Better Option

I think that if the point neighboring to a boundary point have a negative value for their edge, we can make sure that the graph cut cost is minimal, and in this way I think the results can be better.

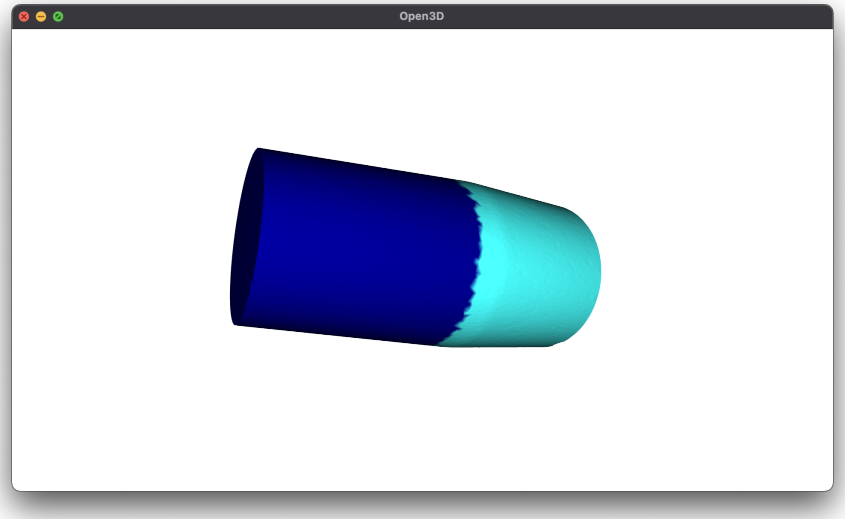

Figure 4: Bipartite Normalized Cut - using Angle

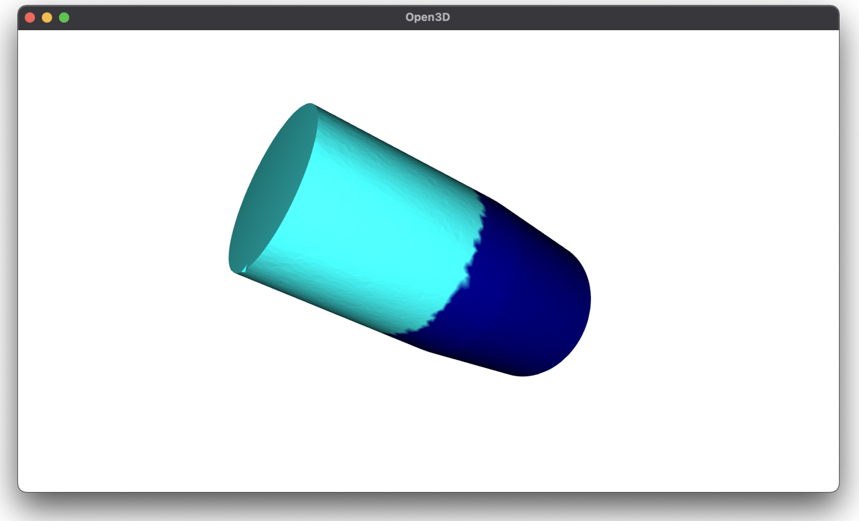

Figure 5: Bipartite Normalized Cut - using Probability

Note: You can view the result of my code in `Results` folder beside the `Q2.py` file. If you wish to reproduce my results, just execute the `Q2.py` file. the scripts saves results from last run into `Results` folder.