Improved Point Cloud Segmentation with Curvature Computation and RANSAC

Talha Rehman Abid

Dakik Yazılım

16 May 2023

Introduction

This report discusses an innovative approach to point cloud segmentation, which employs curvature computation to improve the accuracy of the segmentation process. This strategy has been adopted following a series of trials with alternative techniques, including standard RANSAC, DBSCAN, and PointNet part segmentation, which revealed certain deficiencies in handling specific point cloud scenarios.

Prior Attempts

RANSAC and DBSCAN

The first method attempted was the standard RANSAC (Random Sample Consensus) algorithm, which is a widely adopted approach for fitting a model within noisy data. In conjunction with RANSAC, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), a density-based clustering algorithm, was used to segment the point cloud. However, this approach faced limitations in accurately segmenting points that lay on curves. The algorithm was unable to distinguish between planar and curved points, which led to the misclassification of points lying on the curvature as being part of planar regions. This resulted in poor segmentation, where planes were detected within other planes on the curved regions, significantly impacting the accuracy and quality of the segmentation.

PointNet Part Segmentation

We also experimented with the PointNet part segmentation method. PointNet offers a powerful deep learning framework for handling point cloud data, including segmentation tasks. Nonetheless, PointNet requires semantic and meaningful labels for any given point cloud for effective segmentation. For instance, in an airplane point cloud, labels could include the wings, tail, engine, etc.

However, our data consisted of unique metal parts, for which we didn't have specific, meaningful labels. Moreover, these parts needed to be segmented based on their curvatures and planes, which made the PointNet part segmentation approach unsuitable for our specific use case.

Current Approach

To overcome the limitations encountered with the previous methods, we developed a novel approach that computes curvature for point cloud segmentation. This approach involves multiple steps, described in detail below

1. **Reading and Sampling the Point Cloud:** The first step in our approach is to read a triangle mesh file from a directory and sample points from it uniformly to generate a point cloud. This is accomplished by using the Open3D library's `` `read_triangle_mesh` `` and `` `sample_points_uniformly` `` functions. We then visualize the point cloud using `` `draw_geometries` ``.
2. **Curvature Calculation:** Each point in the point cloud has a curvature associated with it, which represents how much the point deviates from a flat plane. To calculate the curvature, we find the neighboring points, compute their center, and compute the covariance matrix. We perform singular value decomposition (SVD) on the covariance matrix and calculate the curvature based on the singular values. This process is completed using the `` `calc_curvature` `` and `` `compute_curvature` `` functions.
3. **Curvature Threshold Calculation:** With the curvature calculated for all points, we determine a curvature threshold based on a percentile. Points with a curvature above this threshold will be considered as high curvature points. This threshold helps us distinguish between planar regions and regions with high curvature.
4. **High Curvature Point Selection:** We select the points with a curvature greater than the calculated threshold as high curvature points. These points are then visualized in a different color for better distinction.
5. **Point Cloud Manipulation:** To focus on the main segments of the point cloud, we subtract the high curvature points from the original point cloud and add duplicate points. This step is done using the `` `subtract_and_add_point_clouds` `` function.
6. **Outlier Removal:** We remove outliers from the point cloud using the Local Outlier Factor (LOF) method, which measures the local deviation of a given data point with respect to its neighbors. This is done using the `` `remove_outliers_lof` `` function.
7. **Refined RANSAC:** After removing outliers, we apply a refined version of the RANSAC algorithm to further segment the point cloud into multiple segments, each representing a

different surface. This is done using the `` `refined_ransac` `` function. The `` `refined_ransac` `` function uses the standard RANSAC algorithm to iteratively fit plane models to the point cloud data. The points that best fit each plane are grouped together as a segment. The remaining points that are not part of any segment are returned as 'rest'.

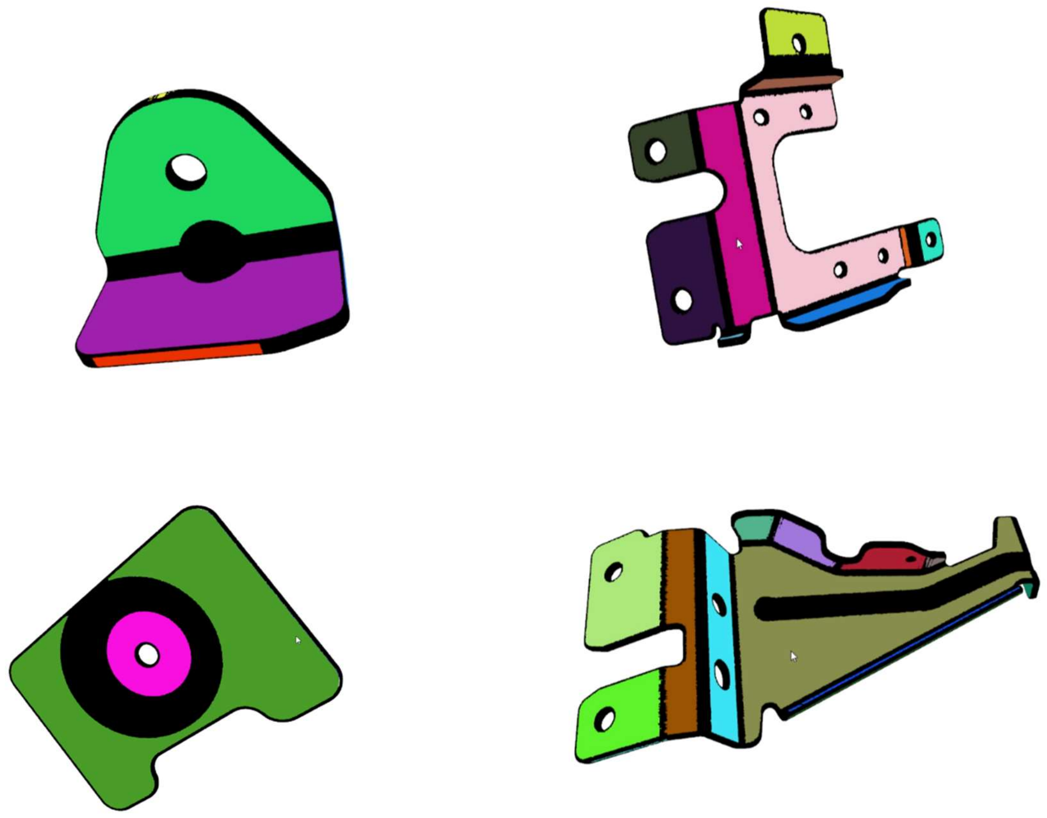
8. **Fine-Tuning RANSAC Results:** After applying the refined RANSAC, we fine-tune the results to ensure the correct identification of each plane segment. This involves using a threshold for the minimum number of points that each segment should have. Segments that do not meet this threshold are re-evaluated for the best fit with other segments. Here, `` `min_points` `` is the minimum number of points that we require for a segment. We set it to 35% of the total number of points in the point cloud. The function `` `assign_colors_to_small_segments` `` iterates over all segments, and for those which contain fewer points than `` `min_points` ``, it tries to merge these small segments with other larger segments. This function ensures that small, potentially noisy, segments do not distort the overall segmentation results. The `` `get_best_distance_threshold` `` function is used within `` `assign_colors_to_small_segments` `` to determine the best distance threshold for merging small segments with larger ones. The resultant `` `updated_segments` `` are then merged into one point cloud a, which is the final point cloud consisting of all the segmented planes with different colors.
9. **Final Visualization:** After all the segmentation steps, the final visualization includes the segmented point cloud, the points identified as having high curvature, and the remaining points that were not included in the segments. This visualization helps to validate the effectiveness of the segmentation. Each segment is painted with a different color for easy identification.

Examples of Point Cloud Part Segmentation

In this section, we present examples of successful and unsuccessful point cloud segmentation to illustrate the effectiveness of our approach and the challenges that can arise.

A. Successful Examples

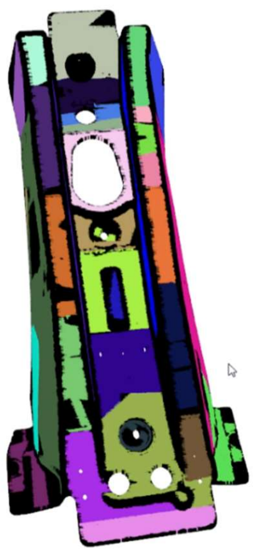
Here, we provide examples where the curvature-based point cloud segmentation method performed well.



Description: These examples demonstrate our segmentation process, wherein the point cloud is color-coded to distinguish various features. Points identified as part of a curvature are marked in black, while the remaining planar regions are color-coded in distinct colors to demarcate different segments.

B. Unsuccessful Example

Despite its effectiveness, our approach can also face challenges. Here, we present an example where the curvature-based point cloud segmentation method faced difficulties.



Description: In this particular case, the metal component exhibited a complex topology, characterized by numerous protuberances, curvatures, boundaries, and isolated planar fragments. As a consequence, our algorithm encountered difficulties in achieving a clean and precise segmentation of the point cloud.

Discussion

The new approach effectively addresses the challenges faced with previous attempts at point cloud segmentation. By calculating the curvature of each point and using it as a critical feature in the segmentation process, we can better distinguish between planes and curved surfaces. This leads to more accurate segmentation and avoids the issues of misclassifying points on curves as planes. The new approach also avoids the need for meaningful labels as required by the PointNet part segmentation method. Given that our data consists of distinct metal parts without any particular semantic meaning, this approach suits our needs better. Instead of requiring predefined labels, we segment the point cloud based on the geometric features of the points themselves, particularly their curvature.

Conclusion

The curvature-based point cloud segmentation approach we have developed provides an effective solution to the challenges we faced in our initial attempts at point cloud segmentation. By calculating the curvature of each point in the point cloud, we can better distinguish between points on planes and points on curves, leading to more accurate segmentation. This approach has proven particularly effective for our use case, where the point clouds represent distinct metal parts without specific semantic labels. As with any algorithm, fine-tuning may be required based on the specific characteristics of the point cloud data. For example, adjusting the percentile used for the curvature threshold or the parameters for outlier removal can help optimize the results. However, the overall method provides a robust foundation for effective point cloud segmentation.

Citations

Poux, Florent, Ph.D. "How to Automate 3D Point Cloud Segmentation and Clustering with Python." Towards Data Science, 15 March 2023, https://towardsdatascience.com/how-to-automate-3d-point-cloud-segmentation-and-clustering-with-python-343c9039e4f5.