Point Cloud *Object* Segmentation Using Multi Elevation-Layer 2D Bounding-Boxes

Tristan Brodeur[1], Hadi AliAkbarpour[1,2], and Steve Suddarth[1]

[1]Transparent Sky, NM, USA
[2]Department of Electrical Engineering and Computer Science, University of Missouri, USA
Email: brodeurtristan@gmail.com, hd.akbarpour@gmail.com, director@transparentsky.net

Abstract

Segmentation of point clouds is a necessary pre-processing technique when object discrimination is needed for scene understanding. In this paper, we propose a segmentation technique utilizing 2D bounding-box data obtained via the orthographic projection of 3D points onto a plane at multiple elevation layers. Connected components is utilized to obtain bounding-box data, and a consistency metric between bounding-boxes at various elevation layers helps determine the classification of the bounding-box to an object of the scene. The merging of point data within each 2D bounding-box results in an object-segmented point cloud. Our method conducts segmentation using only the topological information of the point data within a dataset, requiring no extra computation of normals, creation of an octree or k-d tree, nor a dependency on RGB or intensity data associated with a point. Initial experiments are run on a set of point cloud datasets obtained via photogrammetric means, as well as some open-source, LIDAR-generated point clouds, showing the method to be capture agnostic. Results demonstrate the efficacy of this method in obtaining a distinct set of objects contained within a point cloud.

1. Introduction

Recent advances in photogrammetry applied to wide-area motion imagery (WAMI) systems have enabled the generation of three-dimensional(3D) point clouds that span vasts areas of major metropolitan cities and natural landscapes. These point clouds are usually many millions of points in size, with a low ground sampling distance (GSD) contributing to a dense point cloud with high amounts of detail.

Primary applications for point cloud data include mesh generation, digital surface/elevation model creation, scene understanding, and temporal/structural differencing between two point clouds of the same scene. Of particular interest is the ability to segment buildings, vegetation, and other objects from the scene for classification, mesh-generation, and/or differencing on a per-object basis. For instance, one could classify the type of a single tree that has been segmented from a point cloud (given a low enough ground sampling distance). For meshes, one could have the entirety of a mesh persist within a 3D map and only replace a single object that has changed, thus reducing the need to load the entirety of a mesh when only certain parts of the mesh had actually been altered. Furthermore, 3D models can be applied back to improve WAMI performance and precision. For example, WAMI systems normally require the use of pre-mapped terrain models to project imagery where it belongs. A 3D model generated during a WAMI mission can eliminate many common modeling errors, such as differences between the ellipsoid or geoid used in generating the model and the GPS system in the sensor.

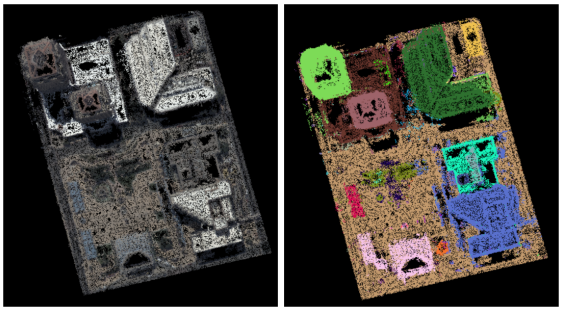

Figure 1: Example of segmentation performed on an input point cloud.

In this paper, we demonstrate a segmentation method that segments *objects* from a scene via the extraction of a

set of 2D bounding-boxes. Note that *object* in this context refers to any physical structure that contains color, texture, geometric, or topological information allowing it to be easily distinguishable from neighboring structures.

A few of the main motivations to finding a 3D bounding box for each *object* within a point cloud are:

1. Plane fitting (via region-growing) to each surface of an *object*.
2. Differencing of various point-cloud datasets via *object*-to-*object* comparisons.
3. *Object* classification.
4. *Object*-based geo-registration error calculation.
5. Digital Surface Model (DSM) / Digital Elevation Model (DEM) generation.

Our method is particularly useful for our WAMI-based surveillance systems. The segmentation process, as well as the underlying 3D models, are generated in a matter of minutes, thus changes can be tracked many times over the course of a mission. Furthermore, the segmentation technique is very effective at segmenting the ground level from "superstructures" that are on top of the ground. The segmented ground elevation (Digital Elevation Model, DEM) is extremely useful for the creation of 2D WAMI imagery. Likewise, the 3D model can be cleaned by applying the following segmentation technique to determine spurious structures that should be removed in the case of generating live 3D views.

2. Related Work

The issue of segmentation applied to 3D point clouds is a well-studied topic with several methodologies presented in the literature. Some approaches to segmentation are meant for planar extraction, while others are meant for extracting instances of objects from the point cloud scene. The methodology of choice may be highly dependent upon one's post-segmentation requirements, available sensor data, and/or time-sensitivity requirements. A general review of point cloud segmentation and classification algorithms is outlined in [4]. Edge-based methods find edges in a point cloud using local surface properties and group points together using the found edges, while model fitting techniques utilize geometric primitives to segment points that conform to the mathematical representation of the primitive. Issues exist with these current methods, however. Edge-based methods are highly sensitive to both noise and point density, while model fitting techniques lose valuable information when segmenting complex shapes in the point cloud.

In the case of autonomous vehicles and/or unmmaned ground vehicles (UGV's), time-of-flight data obtained from LIDAR scanners would be of particular importance in the choice of a segmentation method. Discrimination of objects in the scene based on their distance to the sensor platform is of necessity to meet safety requirements and determine feedback control. An array of segmentation methods that are applied to point clouds obtained from 3D LIDAR scanners are presented and compared in [2]. These methods are defined based on the density of the point cloud obtained from each LIDAR scanner, with grid-based approaches being used for dense data models and interpolation approaches being used for more sparse data models.

For the previously discussed motivations 1 of our WAMI system, both *object*-based segmentation and planar extraction is necessary. Planar segmentation has been widely studied, with initial results being applied to our datasets using the method outlined in [7]. This method uses surface normals and connectivity constraints to find smoothly connected areas in a point cloud. In most of our experiments, this method either over-segments or under-segments planar portions of the scene regardless of the deviation in input parameters r_{th} (residual threshold) and θ_{th} (angle threshold). In a point cloud dataset with a high amount of noise the "noise points" would affect the planar extraction of a side of a building leading to an incorrect geometric interpretation of the building. As well, because we're using a global residual and angle threshold for the entire scene, buildings with smaller angles between planar surfaces were under-segmented due to our thresholds accounting for buildings with high curvature. To assist with planar segmentation, it would be best to first extract each *object* from the scene and then find an appropriate residual and angle threshold for each *object*.

Point cloud instance segmentation has also been widely researched. Machine learning methodologies have been applied to 3D scenes to extract instances of *objects* and semantically label them [12] [13] [5]. In [1], Chen *et al*. extend the U-Net architecture used traditionally in image segmentation [9] to a 3D environment in order to segment buildings, trees, and terrain from several aerial point clouds. Ground pre-processing and post-processing techniques were used to better obtain a correct segmentation of the elevation model, as well as achieve better segmentation results on the remainder of the point cloud model. We have found in our literature review that obtaining a correct elevation model before segmentation of the remainder of the point cloud significantly improved segmentation performance. A recent paper by Lyu *et al*. [6] demonstrates the efficacy in projection of a 3D scene to a 2D plane for the purpose of 3D point cloud segmentation. In the paper the authors project a clustered set of 3D points to a plane from a corresponding graph drawing, and use the U-Net architecture for segmentation of the resultant image embedding. Results show significant improvement over the literature for deep

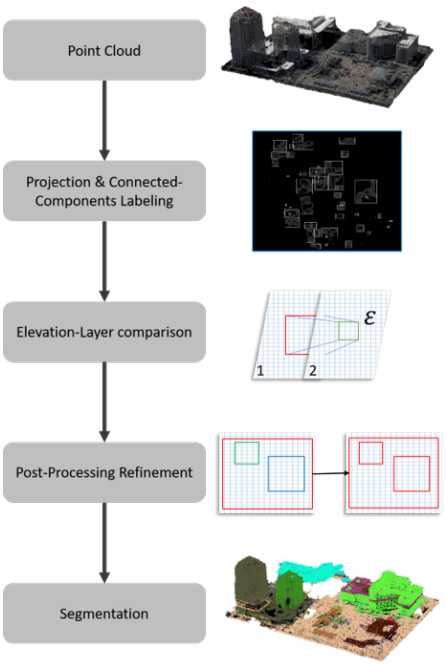

Figure 2: Flowchart of the segmentation algorithm.

3D point cloud segmentation, demonstrating the feasibility for instance segmentation from the 2D image space.

An approach to 3D LIDAR point cloud classification using only the topological information available from the data is presented in [8]. Richter *et al*. begin by applying the region growing approach (method outlined in [7]) to first segment the point cloud, and then classify the point cloud using only the topological data (i.e. connectivity, local flatness, smoothness, and orientation) of each point. Their approach to classification by using topological information is how we approach segmentation.

3. Methodology

Conceptually, the implementation of our algorithm consists of four core steps:

1. Projection of 3D points to a plane.
2. Bounding-box extraction via connected-components labeling.
3. Elevation-layer comparison.
4. Post-processing refinement.

Figure 2 provides a flowchart of the segmentation process.

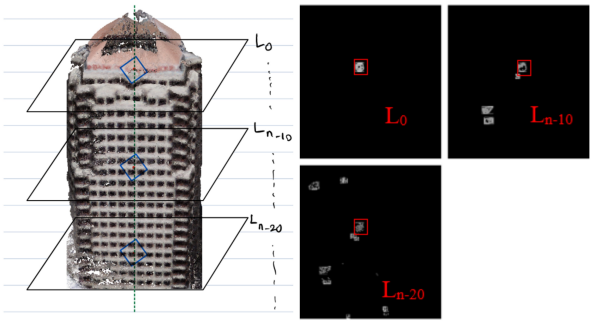

Figure 3: Visual representation of our method. Bounding-boxes are found for the *object* at each elevation layer. L_0, L_{N-10}, and L_{N-10} are layers of the point cloud, with the boxes (in red) showing orthographic projections for the demonstrated building at each layer.

3.1. Orthographic Projection

Finding a 3D bounding-box for each distinct *object* within a point cloud begins with the extraction of 2D bounding-boxes via the orthographic projection of defined layers of the point cloud. We obtain an orthographic projection by "binning" each 3D point to its nearest z-axis integer s.t.

$$\forall p \in P, \lfloor p_z \rfloor = max\{m \in \mathbb{Z} \mid m \leqslant p_z\} \quad (1)$$

where P is the input point cloud, p is a point in the point cloud s.t. $p \in \Re^3$, and m is the set of all integers less than the z value associated with point p. Flooring the z-axis data (as opposed to applying a ceiling operation) was an arbitrary decision and does not affect the end result.

After we bin the 3D points, we create an N x M projection image where N is the resolution of the point cloud in the x axis and M is the resolution of the point cloud in the y axis. The resultant projection image is simply an occupancy grid with a 1 pixel pertaining to a 3D point occupying the x and y coordinates of the projection plane and a 0 pixel indicating free space in the projection plane. Figure 3 provides a visual representation of this technique for a select set of elevation layers of a building.

3.2. Pre-Processing for Ground-Level Extraction

We first apply a pre-processing technique to extract the ground-level elevation data. This step is necessary in pre-processing as this ensures a "stop" limit is placed on the iteration of bounding-box extraction. This "stop" limit is found by simply searching for the largest difference in the number of points between adjacent elevation layers. Extraction of 2D bounding-boxes is complete once the "stop" limit in the iteration is reached, and all 3D points extending below the elevation marker associated with the "stop" limit are considered to be ground-level elevation data. This approach

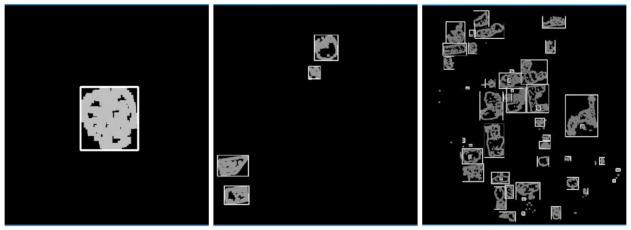

Figure 4: Three elevation layers of the point cloud with bounding boxes surrounding distinct objects at each layer. (Left) Slice of elevation in top-most area of point cloud. (Middle) Slice of elevation in middle area of point cloud. (Right) Slice of elevation in the bottom-most area of the point cloud. Note the dilation of point data of the projection images.

works in most point-cloud datasets where there exists a defined base upon which the *objects* of interest lie. However, this particular ground-level extraction method fails when there exists an *object*-dense point cloud model where most *objects* in the scene have a collection of points at similar elevations (e.g. the roof level of a collection of buildings is the same). In the case of floating artifacts (i.e. objects are captured in a scene but the platform upon which the objects are sitting on is not captured) the approach would also fail. Thus, we assume all input point cloud models will have a unified ground-level upon which all objects of the scene are resting. Note: We state 'unified' because if the ground-level is at different elevations (e.g. point cloud captured of a village on a hill), or the vertical resolution is high enough to capture millimeter differences in ground-level elevation, then this approach would not work. Thus we are deliberating upon different methods for ground-level detection. For the most part, many of the point clouds generated from WAMI systems are of flat terrain areas and so, thus far, we have had no issues pertaining to ground-level extraction when applying this methodology.

3.3. 2D Bounding-Box Determination

To obtain 2D bounding-boxes for a set of 3D points projected on a plane, we must first account for inconsistencies in the spacing between points in the point cloud. To this end, we begin by applying morphological transformations to the projection image. We first apply a dilation operation to the image, effectively increasing the diameter of the points such that neighboring points "touch" one another. An erosion operation is then performed to fill in any holes between points in the kernel. This will assist in removing small "noise" bounding-boxes that would be found if the holes were not to be filled. These "noise" bounding-boxes prove to be an issue in trying to extract a "truth" shape of an *object*.

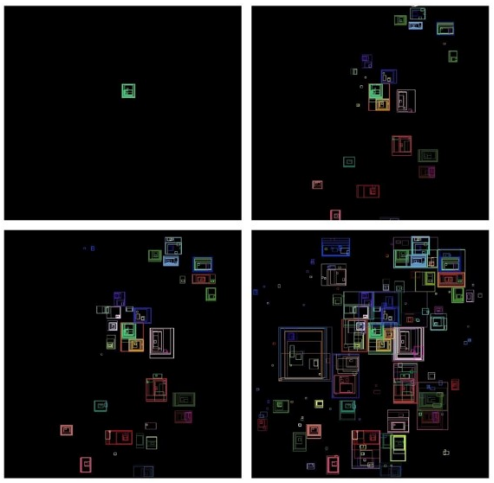

Figure 5: Examples of bounding-boxes being found for elevation layers 76 meters (top-left), 35 meters (top-right), 29 meters (bottom-left), and 15 meters (bottom-right), where the top-most elevation layer is 105 meters and the bottom-most elevation layer is at 0 meters. Each bounding-box is associated with an unique RGB value for the purpose of visualization, where each RGB value denotes the classification of each bounding-box of point-data to its associated object.

The applied operations can be defined mathematically as:

$$I'(i,j) = [I(i,j) \oplus H(m,n)] \ominus H(m,n) \qquad (2)$$

where $\ominus$ is the symmetric difference operator, $\oplus$ is the group addition operator, $I'(i,j)$ is the resultant image at a given index, $I(i,j)$ is the pixel value of the original image at a given index, and $H(i,j)$ is the pixel value of the kernel at a given index. Figure 4 shows a set of *objects* and their associated pixels after morphological transformations are applied.

Note: This is a similar methodology to that of persistent homology techniques to find topological invariants given a defined diameter d around a set of 0-simplices. A static NxN square kernel is preferred in this case because, although the persistent homology techniques proved more fruitful in obtaining true features for a set of points (if one were to record the length of the barcode for a changing diameter d [3]), the trade-off would be computation time. Thus, we settle for a simple set operation and filter out noise features via an array of post-processing techniques.

An appropriate kernel size for the dilation and erosion operations is determined via the resolution of the point cloud. For our purposes, this was determined via the ground sampling distance of the WAMI system. If the resolution

data is not available, we can compute it via the iterative process of checking point neighborhoods and recording the smallest distance:

$$res = min\{\sqrt{((p_x[1] - p_x^k[2])^2 + (p_y[1] - p_y^k[2])^2 + (p_z[1] - p_z^k[2])^2)} \mid \forall p, p^k \in P\} \quad (3)$$

where P is the input point cloud, p is the current point of interest in the point cloud, and p^k is the nearest point to p. Note: p^k is found via a nearest-neighbor search on the k-d tree constructed from the point cloud P.

After morphological transformations are performed, we apply an 8-neighborhood connected-components labeling technique to obtain a collection of orthographically projected *objects*. This method simply checks each of the eight pixel values surrounding a pixel at a given row and column index (i,j) and applies a label l to the pixel if the neighboring pixels are labeled with a label l. Note that the term pixel in this case refers to the orthographic projection of a 3D point at a given elevation layer with the same x and y coordinates. After each pixel in the orthographic projection is labeled, we sort through the labels and extract a bounding-box for each label by finding the minimum and maximum x and y coordinates. A bounding-box is then obtained for a given label l that captures the area of each *object* within the orthographic projection image. We then create an *object* struct to hold necessary information such as the *objects* centroid position in 3D space, the *objects* width and height (determined via the associated bounding-boxes), average area of the bounding-boxes found for the *object*, a vector of all of the bounding-boxes found at each elevation layer of the *object*, a persistence value ϕ (used in post-processing for noise reduction 3.5.1, 3.5.2), a "noise" flag (used in post-processing for noise reduction 3.5.1, 3.5.2), and a UUID assigned to the *object* at its creation.

3.4. Elevation-Layer Comparison

After a set of bounding-boxes is obtained for the set of orthographic projection images I_N, we iterate through the images and "collapse" each image I_i with its prior image I_{i-1}. This "collapsing" process finds the difference between the areas $\Delta\alpha$ and distance between the centroids δ_c of each bounding-box B_k of image I_i to each bounding-box B_j of image I_{i-1}. We define $\|\delta_c\| = \mathrm{d}(B_j.centroid, B_k.centroid)$ and $\Delta\alpha = |B_j.area - B_k.area|$. If the difference is less than the input parameters ϵ_c (centroid threshold) and ϵ_a (area threshold) then we "collapse" the point data from each bounding-box into a single *object*. To ensure smaller parts of an *object* won't affect the "collapsing" process in future iterations, we update an *objects* "area property" with a weighted average of the two areas of the compared bounding-boxes s.t. $object_m.area = w_1 * B_j.area + w_2 * B_k.area$, where the weights are adjusted depending upon the magnitude of the difference $\Delta\alpha$, with a higher weight being applied to the bounding-box with the larger area. Algorithm 1 gives a glimpse as to how these variables relate in the iteration process.

If there does not exist a bounding-box B_k within a given error (ϵ_c and ϵ_a) of the image I_i for bounding-box B_j of image I_{i-1} then we can assume the *object* associated with the bounding-box B_j to be "complete". Once iteration is complete, we can continue on to post-processing for noise reduction. Figure 5 shows the result of the "collapsing" process for an arbitrary selection of elevation layers.

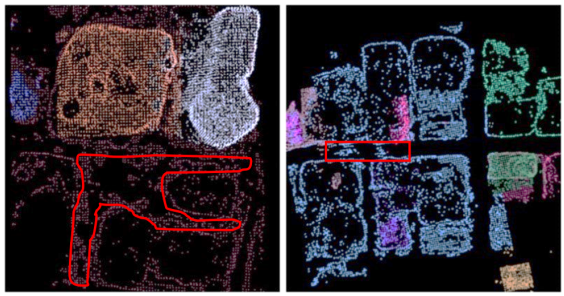

Figure 6: Set of "noise" *objects* with the red border demarcating the set of "noise points". (Left) "Noise" *object* labeled with a maroon color. (Right) "Noise" *object* labeled with a azure color.

3.5. Post-Processing Refinement

Point clouds obtained via photogrammetric means tend to contain a good deal of noise within the model which affects segmentation results. The noise often leads to shared layers of point data for *objects* within a set distance of other *objects* in the scene. These shared points will be labeled to the same *object* (which we term "noise *objects*") thus leading to over-segmentation. Figure 6 shows examples of "noise *objects*" being created as a result of extraneous 3D points. To account for this, we can apply a choice of post-processing techniques to remove noise and/or "noise *objects*" in the model.

3.5.1 Noise Partitioning via the Persistence Metric

To account for the aforementioned issues, we ensure a k amount of *objects* are extracted from a "noise *object*", where k denotes the number of true *objects* within a "noise *object*". To do this, we set a "noise" flag to true in the initial iteration process when one or more bounding-boxes of other *objects* are fully contained within a bounding-box of a newly found *object*. We thus label these *objects* "potential noise *objects*". We iterate through the set of "potential noise *objects*" and check the persistence value ϕ of each *object* encompassed in the "potential noise *object*". Note: the

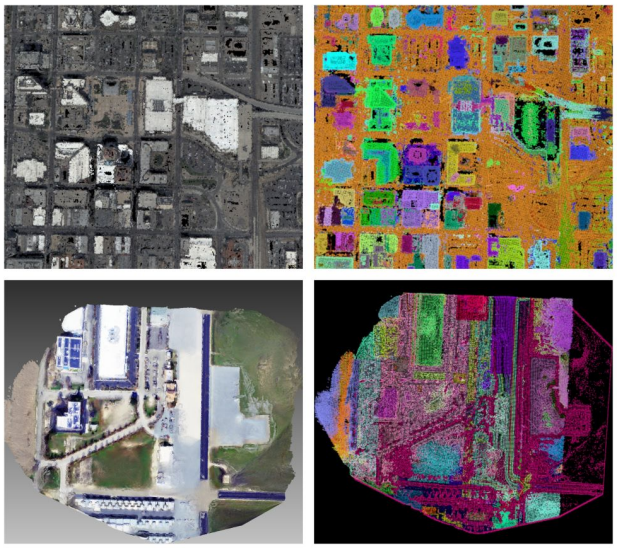

Figure 7: **Qualitative Segmentation results**. (Top-Left) Albuquerque, NM point cloud.(Top-Right) Segmentation result. (Bottom-Left) Paso Robles, CA point cloud. (Bottom-Right) Segmentation result.

persistence value ϕ for an *object* is defined in the initial iteration process as $\phi_{object} = c/N$ where c is the number of orthographic projection images that contain bounding-boxes for the *object*, and N is the total number of orthographic projection images. If the combined set of *objects* has a high persistence value ϕ_{avg} relative to the persistence value ϕ_{noise} of the "potential noise *object*" ($\frac{1}{k}\sum_{i=i}^{k}\phi_i > \phi_{noise}$ where k is the number of true *objects* contained in the "noise *object*), we consider it a "true noise *object*" and segment the "noise *object*" into its k constituent parts. We then iterate through the set of points in the "noise *object*" at the current layer and find the nearest true *object* to partition each point to s.t.

$$\forall p \in \omega, p_{label} = min\{\sqrt{((p_x[1] - p_x^c[2])^2 + (p_y[1] - p_y^c[2])^2)} \mid, p^c \in \xi\} \quad (4)$$

where ω is the set of 3D points contained within the current elevation layer of the "noise *object*", ξ is the set consisting of the centroids for all true *objects* contained within the "noise *object*", and $label$ for a point p in ω is the true *object* to which the point belongs.

Note this does not remove any points, it only partitions the noise data into a set of k true *objects*. To remove a set of "noise points", one may use the following methodology.

Algorithm 1 Segment *objects* from a point cloud

Inputs: Point cloud = {P}, centroid threshold ϵ_c,
area threshold ϵ_a
Output: Set of Bounding-Box Objects {B}
Elevation Projections {E} ← *createProjections* (P)
Bounding-Boxes {B} ← ∅
for $projection$ in E **do**
    $erode(dilate(projection)) \rightarrow projection$
    Current Connected Components {C} ← ∅
    for row, col in $projection$; index = 0 **do**
        label ← ∅
        checkPixelNeighborhood$(row, col) \rightarrow$ label
        if label ∈ C **then**
            (row, col) → C(label)
        else
            (row, col) → label
            C = C ∪ {label}
        end if
    end for
    for cc in C **do**
        Current bounding-box b
        $b_{centroid}$ = ((max_x(cc) - min_x(cc)) / 2,
(max_y(cc) - min_y(cc)) / 2)
        b_{area} = (max_x(cc) - min_x(cc)) *
(max_y(cc) - min_y(cc))
        for bb in B **do**
            Area Difference $\Delta\alpha$
            Average Area α
            Centroid Distance δ
            $\Delta\alpha = |bb.\alpha - b.\alpha|$
            $\|\delta\| = \mathrm{d}(\mathbf{bb_{centroid}}, \mathbf{b_{centroid}})$
            if $bb_{area} > b_{area}$ **then**
                $\alpha = w_1 * bb_{area} + w_2 * b_{area}$
            else
                $\alpha = w_1 * b_{area} + w_2 * bb_{area}$
            end if
            if $(\Delta\alpha < \epsilon_a)\&(\delta < \epsilon_c)$ **then**
                Index $i = indexof(bb, \mathrm{B})$
                $\mathrm{B}_i \leftarrow b$; $\mathrm{B}_i.\alpha = \alpha$
            else
                {B} ← b
                Index $i = indexof(b, \mathrm{B})$
                $\mathrm{B}_i.\alpha = \alpha$
            end if
        end for
    end for
end for

3.5.2 Noise Removal via Bounding-Box Persistence

In the case where removal of "noise points" is preferred, one may (instead of labeling a point to its nearest centroid), "persist" the bounding-boxes from the previous elevation

Dataset	Point Cloud Size	Time
Albuquerque, NM (PMVS)	6,412,701	21 s
Albuquerque, NM (Shell3D)	25,427,513	164 s
Paso Robles, CA	8,551,275	2 s
Brooklyn, NY	2,506,651	24 s
Phoenix, AZ	5,241,988	27 s

Table 1: Datasets and associated run-time.

layer (before the start of a "noise *object*") and remove points not within the bounding-box for a given truth *object*. This runs the risk of removing potentially important information related to the point cloud (such as a bridge connecting two buildings which results in a labeling of sections of two connected buildings as a separate *object*).

3.6. Floating Artifact Removal

Under the assumption that all *objects* in the scene will be subject to the laws of gravity, any *object* that does not have a base or a collection of base *objects* underneath it can be considered "noise" and the set of 3D points associated removed. We determine if an *object* has a base by checking if the area and centroid of the smallest elevation bounding-box for an *object* is contained within the bounding-box of the *object* directly underneath.

4. Results

Figure 7 shows results obtained from running our segmentation method on a set of point clouds.These point clouds (**Albuquerque, NM** and **Paso Robles, CA**) were obtained using photogrammetric means via our WAMI system.

Figure 8 shows results obtained from an open-source point cloud. The **Brooklyn, NY, USA** [10] and **Phoenix, AZ, USA** [11] datasets are point clouds obtained via LIDAR scans.

Table 1 shows the run time of our segmentation algorithm on a collection of point cloud datasets. Note that the number of points does not affect run time averages, but rather the complexity of the scene.

4.1. Results of Varying Vertical Projection Resolutions

The vertical projection resolution is the resolution used for orthographic projection of elevation layers in the scene. We found that increasing the vertical projection resolution (by "binning" points to the set consisting of the set of integers and the midpoint between neighboring integers) results in over-segmentation, wherein each ground-truth *object* within the point cloud contains several layers labeled as belonging to separately labeled *objects*. This is due to inconsistencies in point densities between neighboring elevation layers. Thus, we found that decreasing the vertical projection resolution by "binning" each 3D point's z-axis value to only the set of integers ($\mathbb{Z}$) results in a better segmentation result. This is most likely due to the fact that more information is available at each layer, assisting in finding the true topology of a given *object*.

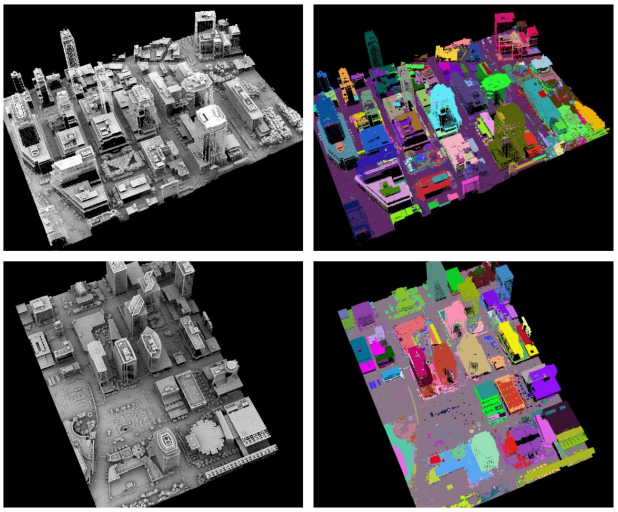

Figure 8: **Qualitative Segmentation results (LIDAR datasets)**. (Top-Left) Brooklyn, NY, USA point cloud. (Top-Right) Segmentation result. (Bottom-Left) Phoenix, AZ, USA point cloud. (Bottom-Right) Segmentation results.

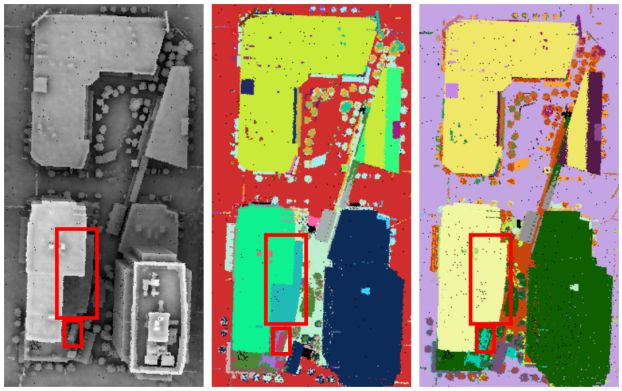

Figure 9: **Results of changing the area threshold ϵ_a.** (Left) Ground Truth. (Middle) Segmentation result of decreasing the area threshold ϵ_a. (Right) Segmentation result of increasing the area threshold ϵ_a.

4.2. Results of Varying Centroid Threshold

Decreasing the centroid threshold ϵ_c assists in better segmentation for datasets with a lower *object* density. Increasing ϵ_c assists in better segmentation for datasets with a higher *object* density. Thus, for datasets such as the **Brooklyn, NY, USA** [10] LIDAR point cloud, where the building density is high, a lower centroid threshold ϵ_c was necessary, so as not to result in under-segmentation due to smaller *object* centroid distances.

4.3. Results of Varying Area Threshold

Figure 9 shows results of altering the area threshold ϵ_a. The red bounding-boxes detail the change in segmentation results. Decreasing the area threshold (as shown in the middle) will provide a more segmented point cloud that classifies the base of the building as a separate *object* from the building on top, while increasing the area threshold results in both the base structure and the building on top being classified as one *object* (as shown on the right). The shade structure (as shown in the smaller red bounding-box) is also classified as a separate *object* with a smaller area threshold. The area threshold ϵ_a may be changed depending on your segmentation needs, however, currently, its largely chosen via a qualitative empirical process.

5. Conclusion and Future Work

A segmentation algorithm for dividing a point cloud into a set of topologically distinct *objects* has been presented. The algorithm uses only the topological information provided from the point data in a dataset. Bounding-boxes are obtained from a set of distinctly labeled components of an orthographic projection image, and a "collapsing" of these bounding-boxes results in an *object*-segmented point cloud. Pre and post-processing techniques are applied to obtain a better segmentation result. Qualitative results are provided showing the effectiveness of this method in obtaining distinct *objects* from a point cloud.

The designed framework for *object* segmentation still has several limitations. First, the choice for the input parameters ϵ_a (area threshold) and ϵ_c (centroid threshold) are empirically obtained, with some prior knowledge on the *object* density (via either qualitative analysis of the point cloud or having knowledge of the ground sampling distance) being required for the centroid threshold, and iteration upon qualitative empirical analysis being required for the area threshold. Second, the vertical projection resolution is hard coded to a projection onto the set of integers ($\mathbb{Z}$). While this has proven to work thus far, the vertical projection resolution should be set via a function of the vertical sampling distance of the underlying point cloud. Third, we would like to obtain quantitative results via a comparison of our segmentation results to hand-labeled, ground-truth datasets. These are all prospective works that would lead to a more robust *object* segmentation result.

6. Acknowledgements

This Work was supported by the National Geospatial-Intelligence Agency (NGA) and the Office of Naval Research (ONR).

References

[1] M. Chen, A. Feng, K. McCullough, P. B. Prasad, R. McAlinden, and L. Soibelman. 3d photogrammetry point cloud segmentation using a model ensembling framework. *Journal of Computing in Civil Engineering*, 34(6):04020048, 2020.

[2] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805, 2011.

[3] R. Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2007.

[4] E. Grilli, F. Menna, and F. Remondino. A Review of Point Clouds Segmentation and Classification Algorithms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42W3:339–344, Feb. 2017.

[5] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020.

[6] Y. Lyu, X. Huang, and Z. Zhang. Learning to segment 3d point clouds in 2d image space. *CoRR*, abs/2003.05593, 2020.

[7] T. Rabbani, F. van den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraints. In H. Maas and D. Schneider, editors, *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, volume 35, pages 248–253. International Society for Photogrammetry and Remote Sensing (ISPRS), 2006. ISPRS commission V symposium : image engineering and vision metrology, ISPRS 2006 ; Conference date: 25-09-2006 Through 27-09-2006.

[8] R. Richter, M. Behrens, and J. Döllner. Object class segmentation of massive 3d point clouds of urban areas using point cloud topology. *International Journal of Remote Sensing*, 34(23):8408–8424, 2013.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[10] V. Võ. USGS LIDAR point cloud of Brooklyn, NY, 2014. Data retrieved from Sketchfab, `https://skfb.ly/6FKGp` under the Creative Commons License `http://creativecommons.org/licenses/by/4.0/`.

[11] V. Võ. USGS LIDAR point cloud of downtown Phoenix, AZ, 2017. Data retrieved from Sketchfab, https://skfb.ly/6GutW under the Creative Commons License http://creativecommons.org/licenses/by/4.0/.

[12] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *arXiv preprint arXiv:1906.01140*, 2019.

[13] B. Zhang and P. Wonka. Point cloud instance segmentation using probabilistic embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8883–8892, 2021.