Estimating 3D Indoor Object Dimensions

Jerry Chen
University of Alberta
jerry3@ualberta.ca

Jonah Chen
University of Toronto
jonah.chen@mail.utoronto.ca

Ruiqing Tian
University of Alberta
ruiqing2@ualberta.ca

Di Niu
University of Alberta
dniu@ualberta.ca

***Abstract*—The rapid development of virtual reality (VR) and 3D sensors call for better algorithms for object localization methods to enhance the user's 3D experience. Size estimation is the problem of predicting the dimensions of an object in a 3D scan, which is necessary for the VR experience as it is difficult to judge the real-world size in a VR environment. Accurate size estimation for the walls and furniture can also expose additional information on virtual tours of houses or museums. Augmented reality (AR) is used widely in indoor tourism and virtual tours. Today, state-of-the-art 3D semantic and instance segmentation models have demonstrated impressive performance for their respective tasks. There are two clustering-based approaches to produce more accurate size estimations from these models' predictions in this paper. One of them can segment and measure the sizes of walls given the points labeled as walls. The other algorithm refines the instance proposals from a segmentation model by removing mislabeled noise. Compared to a baseline method of finding the tightest bounding box of SoftGroup's [1] instance predictions on the S3DIS [2] dataset, with a 2% error threshold, our implementations can achieve $+8\%$, $+6\%$, $+4\%$ higher accuracies on wall length, chair length, and chair width measurements respectively.**

***Index Terms*—3D, Computer Vision, VR, Size Estimation**

I. Introduction

In modern days, tourism no longer has to be physical. The users can either have a real-time remote VR tour or experience a saved scene scanned by LIDAR sensors. As the VR experience and LIDAR sensors capable of capturing 3D scenes in consumer electronics continue to gain popularity, there is an increasing demand for algorithms that perform various 3D scene understanding tasks. Size estimation aims to predict the length, width, and height of an object present within a point cloud. Accurate size estimation is essential for the VR experience because the objects scales in VR scenes can be complicated to perceive by the users.

There has been plenty of prior work on 3D semantic and instance segmentation. State-of-the-art semantic segmentation models like Point Transformer [3] and Stratified Transformer [4], and instance segmentation methods like SSTNet [5] and SoftGroup [1] can achieve impressive results on popular point cloud datasets like S3DIS. However, a high average precision (AP) in instance segmentation does not directly lead to accurate size estimation. A few mislabelled points far from the object have minimal impact on the AP but lead to a significant error on the object's dimensions shown in Figure 1. Furthermore, the current state-of-the-art instance segmentation methods are based on clustering and perform poorly on objects adjacent to each other.

978-1-6654-5888-7/22/$31.00 ©2022 IEEE

PointNet++ [6] is a commonly used backbone for processing 3D point clouds. It can generate point-wise and global features from the point cloud for downstream tasks. A common approach is to predict objects' bounding boxes directly from a 3D point cloud using its PointNet++ features. VoteNet [7] is a model that uses point features from PointNet++, and cluster to generate instance proposals, then use these instance features to generate bounding boxes(location, scale, and rotations). A succeeding work is ImVoteNet [8] which adopts a similar structure to VoteNet, but ImVoteNet also uses object mask prediction on 2D images for each scene to enhance instance proposals from 3D scenes.

Other than 3D object measurements, estimating the dimensions of flat surfaces such as walls also has practical applications. For example, when people are planning to move into a new place, they may not be able to visit in person. Then, they can have a 3D scan of the rooms and determine their wall sizes. This information can help them decide how much paint or wallpaper they need to decorate the walls without visiting or bothering the seller to measure the dimensions manually. Another use case would be to determine a room's size since having dimensions for each wall segment allows one to calculate the size of the room.

In this work, we propose two clustering-based approaches for size estimation. For flat objects like wall segments, we leverage the predictions from a semantic segmentation model to obtain the points belonging to the walls. Next, we use clustering algorithms to form raw wall segments, including corner handling and noise removal via rule-based methods. Then, we first estimate the slope at each point by performing linear regression on nearby points. For other objects, like tables and chairs, we leverage predictions from an instance segmentation model and perform distance-based clustering. This implementation allows us to correct some mislabeled points in instance segmentation to generate more accurate size estimates.

We evaluate our method on the S3DIS dataset using raw results from SoftGroup as the baseline, and we can show the improvements of our post-processing method.

II. Methods

A. Problems

In this paper, we solve the issue of measuring object sizes in various rooms given their 3D point clouds. Each object

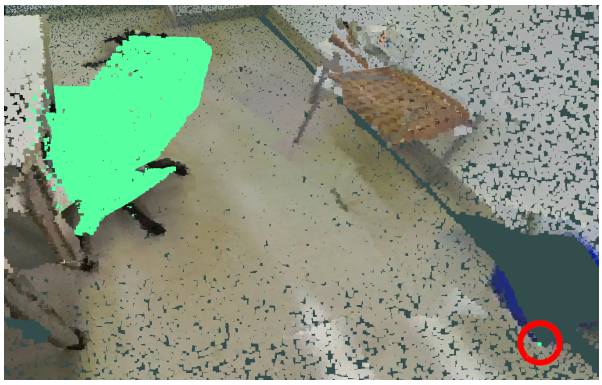


Fig. 1. A chair in the scene is detected and labeled green. There is a mislabeled point in the bottom right of the image (highlighted with a red circle). This one dot will not have much impact on AP evaluation, but it will create a bounding box completely different from the ground-truth bounding box.

is seperated into two categories, wall segments and single objects such as furniture. There are machine learning-based models to predict the point labels and get all points in walls, but the issue is how to divide it into each wall segment. For single objects, we also used models to predict their labels and instance masks. Hence, for each object, we can identify the set of points belonging to it. However, these predictions may not yield very accurate dimensions; therefore, we will employ some post-processing techniques to improve the prediction quality.

B. Wall segments measurements

In a 3D point cloud of the entire room P with the x, y, z coordinates containing N points. We adopt a model that can take P and produce $L = \{l_1, \ldots, l_N\}$, such that each l_i is the class label of the i^{th} point. These labels indicate the type of points such as the wall, floor, table, chair. Using the labels, we filter out the points labeled as wall P^w.

Once we have the P^w, we need to segment the P^w into wall segments. To do this, we need to identify how each wall segment differs from the others. The relevant feature is the rotation of each segment, in other words, the slopes from the top view. To approach this, for every point p_i^w in P^w, we would find neighbor points—the set of points in a region of interest (ROI). Then we fit a line through the points in the ROI whose slopes represent the rotation of wall segments. We convert the slopes into angles using the $\arctan$ function. The angles a_i^w from the line is assigned to p_i^w.

The next step would be clustering the points by their angles. Intuitively, the points with similar angle either belong to the same wall segment or to parallel wall segments. Then, we perform another level of clustering based on x, y coordinates for each cluster found previously. This way, we can separate the parallel walls with the same slopes. Lastly, we calculate the tightest bounding box for each separated cluster (wall segment) and measure the sizes of the bounding boxes to obtain the wall's dimensions.

This method seems good on paper, but it has difficulty handling corners. When finding the best fit line at each corner, the line will pass through both wall segments instead of the line right on a wall segment. To solve this issue, we will find the sum of the squared scaled distance of each point in the ROI to the best fit line. If the ROI only contains one wall segment, the value should be small, and if the ROI is at the corner, this value will be large because most of the points do not lie on the line. Once we identify all the corners, we can conclude that the other clusters are well-identified wall segments but missing some points near the corners.

The next step is merging the corner points back to corresponding wall segments. For each identified wall segment, we calculate the best fit line for the entire wall segment. If a corner point lies on any of the segment's lines and is within a reasonable distance to the wall segment, then it can be assigned to that wall segment. If a point lies on multiple segments' best fit lines, that point will be assigned to the closest wall segment. Lastly, if a point does not lie on any wall segment's best fit lines, it is treated as noise and discarded.

Finally, we run an algorithm that calculates the tightest bounding box on points from each wall segment to estimate its size. Fig. 2 shows a flow chart of the algorithm.

Algorithm 1: Find Wall Segments

Input : P^w - 3D points labeled as wall
Output: $wallSegs$ is a list of list of points, each inner list represents a wall segment

1 $A^w \leftarrow$ LinearRegression angle for each p in P^w
2 $P^c \leftarrow$ DetectCorners(P^w, A^w)
3 $P^{nc} \leftarrow P^w \setminus P^c$
4 $C^A \leftarrow$ ClusterByAngle(A^w, P^{nc})
5 *wallSegs* $\leftarrow$ empty
6 **for** *cluster*$_i$ *in* C^A **do**
7 *wallSegs*.insert(ClusterByXY(cluster$_i$, P^{nc}))
8 **end**
9 **for** *p in* P^c **do**
10 insert p to the wallSeg that has best distP2Line(p,wallSeg) and distP2Slope(p,wallSeg)
11 **end**
12 **for** w_i,w_j *in wallSegs where* $i \neq j$ **do**
13 **if** $dist_{slope}(w_i,w_j) \leq T^{slopeDist}$ *and* $dist_{segs}(w_i,w_j) \leq T^{segmentDist}$ **then**
14 Merge w_i and w_j in wallSegs
15 **end**
16 **end**
17 **return** wallSegs

In algorithm 1, lines 1 to 3 pick out the poorly fitted points as corners and find the slope for each point using its neighbors. Then in lines 4 to 8, we perform two levels of clustering on points not labeled as corners. First, we cluster based on each point's slope. Then, we cluster again based on their x, y coordinates. In lines 9 to 11, we let each point in the

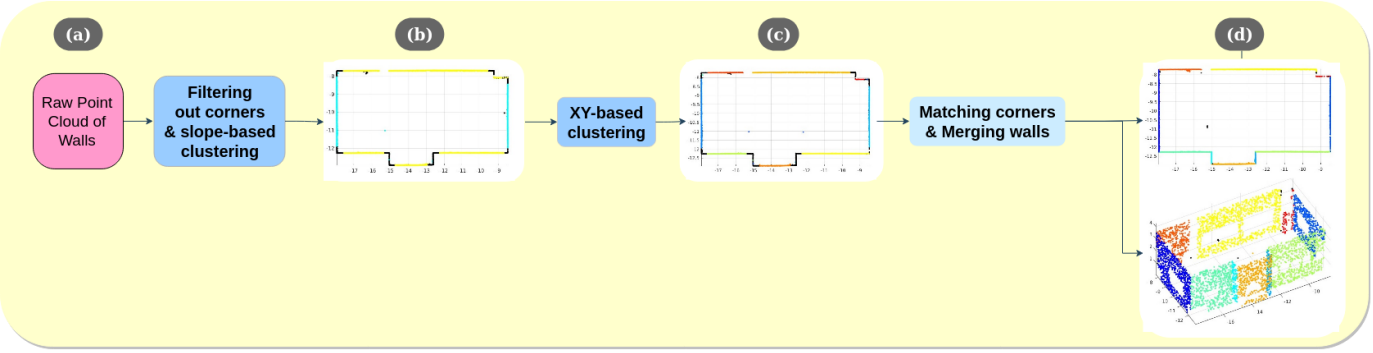


Fig. 2. (a): Point cloud of only walls generated from raw S3DIS point cloud data and the semantic prediction from SoftGroup. Based on the point clouds from raw data, calculating slopes at each point and masking out the corners; clustering based on the slope of each wall segment. (b): Identifying the corner points (black) and clustering based on the slopes of normal points (cyan, yellow). (c): Clustering by x, y coordinates using non-corner points. The corner points are black. Other colors represent different clusters (wall segments). (d) Top view and 3D view of the predicted wall segments or surfaces after corner points are re-assigned, and appropriate wall segments are merged.

corners choose the best wall segment to join. Lastly, in lines 12 to 16, we compare the pairs of wall segments by distance and their slope similarity. If both are below their respective thresholds, we can conclude they are the same piece of wall and merge the clusters; otherwise, keep them the way they are. In the end, the wall clusters left are the final wall segment predictions.

C. Single object measurements

Similar to the wall measurement section, we also start with P and L, but we also have $I = \{i_1, \ldots, i_N\}$ such that each point has an instance ID assigned from an instance segmentation model. This time, we only consider the points labeled as indoor objects such as tables or chairs. The simple approach here is to find the tightest bounding box for every set of points with the same instance ID.

However, these models are generally trained and evaluated for point-wise accuracies, not space-wise accuracies. It means that if a prediction of the points belonging to an object has some noise, such that there are a couple of mislabeled points far away from the object, the evaluation will grant high accuracy due to a small number of mistakes compared to a large number of correctly labeled points. However, it will result in terrible bounding box intersection over union (IoU) results.

As previously mentioned, there are issues where some mislabeled points of an instance are too far from the actual object, resulting in the wrong shape. Intuitively thinking, if a few points are far away from the majority points, they are likely to be outliers or noise. As most points are connected, we use clustering to find the outlier points. We remove these outlier points to clean up the instance masks and generate more accurate bounding boxes.

III. Evaluation

A. Dataset

S3DIS: this dataset contains scans from multiple rooms. Each scan is a dense 3D point cloud with RGB values for each point. All scenes have both point-wise and instance-wise semantic labels. S3DIS contains over 200 scans of primarily meeting rooms, offices, and hallways. The labels contain objects including tables and chairs, as well as walls, floors, and beams.

B. Metrics

There are two main evaluation metrics we will use for wall segments and one additional metric for other objects. The first metric is the percentage error for each dimension of the objects, which is calculated by finding the percent differences of each pair of dimensions of the corresponding prediction and ground truth object. This metric indicates how well our method predicts the sizes. The second metric is instance-wise precision and recall. We have many merging and removing operations in our algorithm, so it is important to determine the number of instance prediction mistakes. For example, recall metric can tell us if we mistakenly removed any instances as noise. The last metric for objects other than walls is bounding box IoU in 3D space, which can describe the amount of overlapping between prediction and ground truth.

C. Procedures

In our experiments, we use SoftGroup, one of the state-of-the-art instance segmentation methods, for our semantic and instance predictions because of their easy-to-use published code. Despite not being a top semantic segmentation model today, SoftGroup still achieves 68.9% overall mIoU and 85.3% mIoU for walls on the S3DIS Area 5 validation. Although the Stratified Transformer is capable of 72.0% mIoU on S3DIS, it does not predict instance masks so we cannot use it as a baseline.

1) Wall Evaluation: The *baseline* results for wall size estimation are calculated by finding the tightest bounding boxes directly from SoftGroup instance proposals. The *filtered* results are calculated similarly to baseline, except we do clustering using DBSCAN [9] with $\varepsilon = 0.1$m on each proposal from SoftGroup to remove the noise.

TABLE I
WALL SIZE ESTIMATION EVALUATION ON S3DIS

Method	Precision	Recall	MRE(L)	MRE(H)
Baseline	80.7%	79.3%	10.0%	4.8%
Filtered	83.7%	82.5%	**9.8%**	4.5%
Our Method	**84.5%**	**91.5%**	12.0%	**3.2%**

For our method, we used a ball of radius 0.3m about each point as the neighborhood for linear regression. We use DBSCAN as our clustering method. For step one, we used $\varepsilon = 15\,\mathrm{deg}$ and a minimum of 50 points per cluster. For step two, we used $\varepsilon = 0.1$m and a minimum of 10 points per cluster. In addition, since step one clustering is based on the slope of each wall segment, some wall segments have slopes with some extreme values, like infinity, will be complicated to handle. For those cases, a rotation is applied to wall segments and rotated back after clustering.

We first retrieve the set of points belonging to the walls from the semantic segmentation model. We randomly sample 10000 points from this set as input to our method for finding the wall segments. For the baseline method, we use the wall instance predictions from the SoftGroup directly as the wall segments. And we only use the largest cluster from DBSCAN with $\varepsilon = 0.1$m on SoftGroup's instance predictions in the filtered method. This serves as a way to filter outlier points for better size estimates. Note that SoftGroup achieves 72.6% AP_{50} on the walls in the S3DIS Area 5 validation.

After determining which points belong to each wall segment, we calculate the tightest bounding box around each wall segment. If the bounding box has a width below 0.1m, we widen the bounding box to 0.1m for the box matching step. We use the same procedure to process the ground truth bounding boxes. Then, we match the ground-truth bounding boxes to the predicted boxes by descending IoU values on every pair. Finally, we record those matched boxes for precision and recall calculations and calculate the difference between corresponding dimensions for error rates.

2) Single Object Evaluation: Similar to the wall evaluation, the *baseline* method directly computes the tightest bounding box on SoftGroup's instance predictions. For our implementation, we also use SoftGroup's instance segmentation. Then, we perform clustering using DBSCAN with parameter $\varepsilon = 0.1$, and we filter out all points that do not belong to the largest cluster. After clustering and generating the refined instance masks, we calculate and measure the tightest bounding boxes from the ground truth and refined instance labels for comparisons. The matching method is similar to the wall evaluation part: we match the ground-truth boxes to the refined prediction boxes and keep the pairs in descending IoU score order. Lastly, we calculate the bounding box IoU, precision, and recall alongside the relative errors for each dimension's measurement.

D. Results

1) Wall: Table I shows the precisions, recalls, and the mean relative errors (MRE) for the length (L) and height (H) of the wall segments. Our proposed method demonstrates stronger classification performance than the baseline having a 3.8% higher precision and 12.2% higher recall. Better performance means our method is effective in removing false proposals, as well as matching found segments. The lower MRE in length prediction comes from problematic samples that would cause significant percent errors. For example, the sample3 contains multiple pieces of walls in ground-truth, and from it, we can see a couple of things that would cause issues. A wall is broken by something like doors or windows labeled as one wall in the ground truth, but our method would predict the two pieces as two separate wall segments. Another issue is that a piece of wall between two corners may not be detectable by our method due to its small size. For more meaningful evaluation, we calculate cumulative distribution function (CDF) values for the dimension errors in table II. The percentage in the top row indicates the error threshold. For example, <1% with 0.63 of length in our method means 63% of our predictions have an error less than 1%. This table indicates that our method can generate better predictions than the overall in SoftGroup, without disturbance from corner cases. Aside from accuracy measures, we also measured the time required to run the process. On average of 128 scenes with 10000 sampled points each, the program will take 6.03 seconds to finish on Intel CPU i9-9900X. However, a large amount of time (2.40 seconds) is consumed by calculating point angles due to looping over every point without optimization. In real-world applications, this step can be replaced by using the 2D point norms from the camera.

TABLE II
CDF FOR DIMENSION ERRORS ON WALL SEGMENT MEASUREMENTS

	Method	<1%	<2%	<5%	<10%	<30%
Length	Baseline	0.51	0.65	0.75	0.81	0.86
	Filtered	0.52	0.66	0.76	0.82	0.86
	Our Method	**0.63**	**0.73**	**0.81**	**0.85**	**0.90**
Height	Baseline	0.61	0.78	0.86	0.88	0.97
	Filtered	0.60	0.78	0.86	0.89	0.97
	Our Method	**0.72**	**0.85**	**0.91**	**0.94**	**0.98**

2) Single Object: Of the thirteen classes in the S3DIS dataset, the chair, table, bookcase, and sofa are the four classes describing standard 3D objects, as the others are either flat like board and ceiling; or miscellaneous like clutter. Hence, our evaluation is performed on these four classes using the same metrics as the wall evaluation, along with bounding box average IoU and mean relative error on width. Our method shows a significant improvement in box average IoU, length, and width errors, especially for chairs. A large amount of noise present in the instance masks that are far from the object can be filtered out by our method(figure 1). Removing the noise results in tighter bounding boxes and eventually lead to more accurate size measurements. However, the improvement for height measurements is minor as we can not cluster on the Z coordinate because thin objects like the legs of tables and chairs are not captured well in S3DIS. Also, our method only achieves a slight improvement

TABLE III
SINGLE OBJECT SIZE ESTIMATION EVALUATION ON S3DIS

Object	Method	Precision	Recall	Avg. IoU	Mean Relative Error		
					Length	Width	Height
Chair	Baseline	92.7%	90.8%	71.9%	25.2%	8.7%	8.4%
	Our method	**92.8%**	**91.0%**	**76.2%**	**12.7%**	**6.6%**	**8.2%**
Table	Baseline	87.4%	86.3%	66.1%	17.5%	15.2%	23.0%
	Our method	**87.6%**	**86.5%**	**66.2%**	**17.0%**	**13.8%**	**22.9%**
Bookcase	Baseline	83.5%	74.1%	62.2%	23.0%	24.2%	15.6%
	Our method	**83.6%**	**74.2%**	**63.0%**	**22.5%**	**22.6%**	**15.5%**
Sofa	Baseline	94.2%	87.5%	78.5%	**16.6%**	12.9%	4.3%
	Our method	94.2%	87.5%	**79.8%**	16.7%	**6.6%**	4.3%

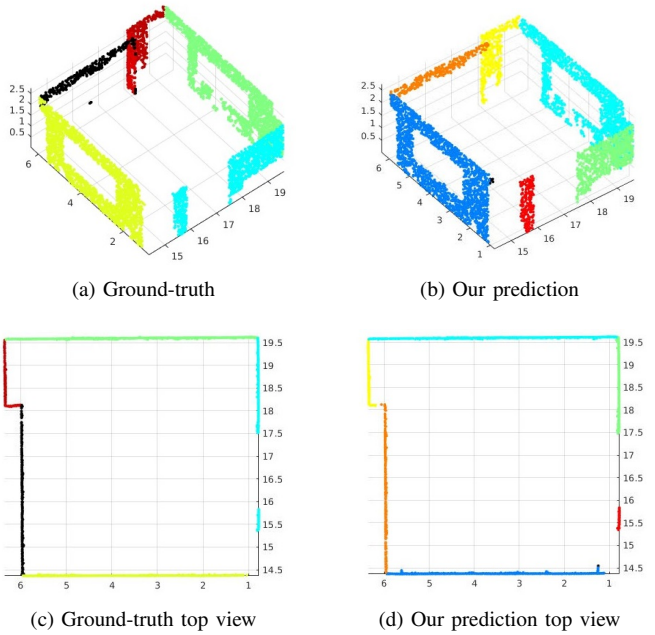
TABLE IV
CDF FOR DIMENSION ERRORS ON SINGLE OBJECT MEASUREMENTS

	Method	<1%	<2%	<5%	<10%	<30%
Chair (Length)	Baseline	0.30	0.39	0.55	0.69	0.82
	Our method	**0.33**	**0.43**	**0.61**	**0.77**	**0.91**
(Width)	Baseline	0.35	0.44	0.62	0.78	0.93
	Our method	**0.37**	**0.48**	**0.64**	**0.80**	**0.96**
(Height)	Baseline	0.38	0.58	0.75	0.80	0.86
	Our method	**0.39**	**0.59**	0.75	0.80	0.86
Table (Length)	Baseline	0.39	0.46	0.61	0.69	0.83
	Our method	0.39	0.46	0.61	**0.71**	**0.84**
(Width)	Baseline	0.32	0.42	0.60	**0.73**	0.87
	Our method	**0.33**	**0.43**	**0.61**	0.72	0.87
(Height)	Baseline	0.13	**0.21**	0.46	0.69	0.84
	Our method	0.13	0.20	**0.48**	**0.70**	0.84
Bookcase (Length)	Baseline	0.18	0.30	0.49	0.63	0.79
	Our method	**0.19**	**0.31**	**0.50**	**0.64**	**0.80**
(Width)	Baseline	0.13	0.22	0.41	0.59	0.79
	Our method	0.13	0.22	0.41	0.59	**0.80**
(Height)	Baseline	0.27	0.36	0.51	**0.69**	0.86
	Our method	0.27	0.36	0.51	0.68	0.86
Sofa (Length)	Baseline	0.45	0.58	0.63	0.68	0.83
	Our method	0.45	0.58	0.63	0.68	0.83
(Width)	Baseline	**0.33**	**0.48**	0.73	0.80	0.93
	Our method	0.30	0.45	0.73	0.80	**0.95**
(Height)	Baseline	0.23	0.48	0.78	0.93	0.98
	Our method	**0.25**	0.48	0.78	0.93	0.98

in precision and recall because we only refine SoftGroup's instance proposals for single objects rather than generating new ones. In addition, we measured the time for refining objects. There were 400k points on average for several scenes, and the time taken is about 4.69 seconds per scene.

IV. EXPERIMENTS ON PHYSICAL ARTIFACTS

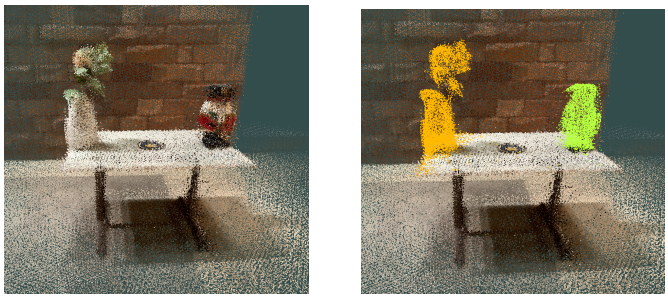
We test the effectiveness of SoftGroup on untrained objects using some artifacts shown in figure 4(a). The nutcracker and vase are not part of the training set of the S3DIS dataset, so we expect them to be labeled as "miscellaneous". We captured the scans using an iOS application called *3D Scanner App* on a fifth-generation iPad Pro. These scans are likely to have different distributions on 3D data. The objects are well-segmented when they are part of a reasonably large scan. So, we can adapt our method to identify and refine the small objects' instance masks. Then we use instance prediction results from SoftGroup to measure the predicted the object sizes using the baseline method and our method. For the ground truth, we measured the vase and nutcracker



(a) Ground-truth (b) Our prediction

(c) Ground-truth top view (d) Our prediction top view

Fig. 3. Images (a) and (b) show the broken wall issue. In the Ground-truth, the two blue pieces belong to the same wall segment, but we treated them as two segments with red and green. Images (c) and (d) show a piece of wall at the top section of the left side(horizontal bright red wall segment) that is missing in our prediction due to its small size between the corners



(a) Original scene (b) Prediction on scene

Fig. 4. (a) is the original scene of both artifacts on the table. (b) have both artifacts labeled as distinct objects by the SoftGroup instance segmentation model.

TABLE V
MEASUREMENTS FROM NUTCRACKER AND VASE

	Method	Length	Width	Height
Nutcracker	Ground Truth	0.26	0.19	0.45
	Baseline	0.48	0.30	0.47
	Our Method	0.27	0.24	0.47
Vase	Ground Truth	0.41	0.25	0.73
	Baseline	0.87	0.80	0.77
	Our Method	0.80	0.35	0.77

manually. From those measurements, we demonstrate that our implementation can improve the results of SoftGroup predictions. However, the lengths of the vase predicted by SoftGroup and our method predictions are erroneous due to misclassifications. In figure 4(b), a patch of the table beside the vase is also classified as the vase, and part of this patch is well connected to the vase that our clustering algorithm cannot eliminate it as noise.

V. CONCLUSION

We propose two clustering-based algorithms to determine the sizes of 3D objects. The first algorithm predicts wall segments directly from point labels using clustering on the points' rotation angles. The second algorithm predicts the objects' sizes by refining instance proposals and removing noise from the model prediction. From the results of our evaluation, we can conclude that our methods can generally produce more accurate size estimations than the baseline. However, the results still heavily rely on predictions from the semantic or instance segmentation model; therefore, if the model makes any huge mistake, our method cannot guarantee to fix it. A common cause of the problems is when two or more objects with the same label are directly adjacent to each other. SoftGroup has trouble separating them, causing them to be treated as one single object. Overall, the proposed algorithms can improve the VR quality, enhancing the distant tourism experience.-++-+———-+

REFERENCES

[1] T. Vu, K. Kim, T. M. Luu, T. Nguyen, and C. D. Yoo, "Softgroup for 3d instance segmentation on point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 2708–2717.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1534–1543.

[3] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[4] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia, "Stratified transformer for 3d point cloud segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8500–8509.

[5] Z. Liang, Z. Li, S. Xu, M. Tan, and K. Jia, "Instance segmentation in 3d scenes using semantic superpoint tree networks," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2763–2772, 2021.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[7] Z. Ding, X. Han, and M. Niethammer, "Votenet: A deep learning label fusion method for multi-atlas segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 202–210.

[8] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "Imvotenet: Boosting 3d object detection in point clouds with image votes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4404–4413.

[9] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.