REIS: A Visual Analytics Tool for Rendering and Exploring Instance Segmentation of Point Clouds

Pedro S. de Freitas*†, Wilson Gavião†, João Comba* and Cláudio R. Jung*

*Institute of Informatics, Federal University of Rio Grande do Sul

† SENAI Innovation Institute for Integrated Solutions in Metalmechanics

***Abstract*—3D Instance Segmentation (3DIS) of Point Clouds (PCs) is valuable for applications like autonomous vehicles, robotics, and Building Information Modeling (BIM). Current work on this topic is guided mainly by global metrics like *mAP*, which arguably do not support a deep, informed analysis of technique tradeoffs and, more importantly, directions for improvement. Qualitative analysis is widely adopted to provide such guidance, but it is generally implemented ad-hoc. This is true across many tasks in Deep Learning, but PC 3DIS is especially challenging to visually analyze due to the many variables involved: three spatial dimensions, colors, semantic labels, and instance IDs. We propose REIS, a visual analytics tool for Rendering and Exploring Instance Segmentation results. It supports qualitative analysis in two ways: first, through PC renderings targeted at efficient investigation of 3DIS results; second, by providing a systematic way to explore these results via the interactive Instance Detection Matrix- a confusion matrix analog that summarizes error and success cases, and allows the user to navigate through them. To show the efficacy of REIS, we use it to evaluate a state-of-the-art 3DIS approach on the S3DIS dataset. Our code is available at https://github.com/pedro-sidra/pcloud_explorer.**

I. Introduction

Point Clouds (PCs) are a natural representation of acquired data for many sensors, such as LiDAR and stereo/depth cameras. With the rising usage of these data sources for many applications, such as autonomous vehicles, robotics, and Building Information Modeling (BIM) [1], intelligent systems that can process PCs have also become more valuable.

Processing and interpreting PCs typically involve two problems: semantic and instance segmentation. Semantic segmentation aims to predict a label $l \in \mathbb{K}$ for each point, where $\mathbb{K}$ is a finite set of classes. In 3D Instance Segmentation (3DIS), the goal is to classify each point in this way and further isolate each object instance. This goal is achieved by associating each point in the scene with a unique instance identifier (instance ID). In this case, the semantic label is often described instance-wise, i.e., points within the same instance have the same label.

A standard metric to evaluate and compare 3DIS methods is the mean Average Precision (mAP), which summarizes the compromise between precision and recall considering a given Intersection-over-Union (IoU) threshold (typically 0.25 or 0.5) [2]. Since 3DIS is a superset of semantic segmentation, the mean Intersection-over-Union ($mIoU$) can also be reported to represent the semantic-level segmentation quality [3].

Although these metrics provide a way to analyze results at large scales and quantify progress, recent works have been critical to the blind use of mAP [4], [5]. In particular, some argue that focusing on global mAP might hide important aspects of the results, such as tradeoffs between different categories of errors [6]. It has also been shown that mAP can be "gamed" to increase the metric by essentially "hedging" predictions, a characteristic of mAP that can mislead the community into optimizing for the wrong objective [5].

As an alternative, recent works proposed more detailed result measurements. The TIDE framework introduced in [6] breaks down the mAP into different components, each relating to a type of detection mistake and allowing a more in-depth analysis of object detection metrics. Regardless of the chosen quantitative metric, qualitative analysis is still paramount for identifying error cases. However, simultaneously analyzing many variables creates challenges to this type of in-depth analysis in 3DIS. The user must inspect three spatial dimensions, plus colors, semantic labels, and instance IDs (the latter two having predicted and ground truth variations).

This work proposes a visual analytics tool for Rendering and Exploring Instance Segmentation (REIS) of PCs that supports qualitative analysis of inference results. We introduce a *Instance Detection Matrix* that supports highlighting, locating, and visualizing different types of errors. The core of the system is the 3D rendering view, which renders a PC augmented with instance segmentation results, mapped into visual cues. We use REIS to study the performance of a state-of-the-art (SOTA) 3DIS approach [2] on the S3DIS dataset, summarizing the pros and cons of the method.

II. Related Work

Data in Point Cloud 3DIS Datasets: a comprehensive review about PC datasets is given in [7] and [3], which highlight the main datasets used for the 3DIS problem - S3DIS [8], ScannetV2 [9] and, more recently, STPLS3D [10]. We represent PCs in a tabular format with n lines and m columns, where n is the total number of points and m is the number of point attributes. Each column is represented as a vector with size n, and common attributes among all datasets are: the coordinate values $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$; the point colors $\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}$; the semantic labels $\boldsymbol{L}$, with $L_i \in \mathbb{K}$, and $\mathbb{K}$ is the dataset-dependent set of classes; and the set of instance ID's $\boldsymbol{I}$.

This study was financed in part by CNPq and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

979-8-3503-3872-0/23/$31.00 ©2023 IEEE

Output Data Produced by Point Cloud Instance Segmentation: PC instance segmentation methods generate the predicted columns $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{I}}$, where the former is the predicted semantic label, and the latter is the predicted instance ID. Particularly, $\hat{\boldsymbol{I}}$ does not need to numerically match the ground truth (GT) instance IDs in $\boldsymbol{I}$, but should instantiate objects in such a way that they have significant IoU with the GT objects.

Current SOTA approaches for PC 3DIS generate different kinds of output data, depending on the chosen strategy for generating the instances. The two main variants of these strategies are currently the "Top-Down" and "Bottom-Up" paradigms [11]. Top-down methods start with an object detection or instance proposal strategy, and process them to generate the instance IDs $\hat{\boldsymbol{I}}$ along with the corresponding instance-level categories [12], which are propagated to obtain the point-level semantic labels. Bottom-up methods obtain the per-point semantic labels $\hat{\boldsymbol{L}}$ and then cluster them, along with other point-level features, to obtain object instances [13], for which the semantic label is recomputed/refined. Hence, bottom-up methods might generate both point-wise and an instance-wise semantic label, which might not be the same [2].

Related visualization/rendering systems: since PCs are a ubiquitous format, general-purpose software can be used for visualization [14], [15]. These tools can be used to investigate 3DIS results through features such as color mappings, but are not targeted to this task. On the other hand, the robotics and autonomous driving communities have developed many targeted tools to visualize PC results [16], [17]. These are, however, scoped at specific applications, reducing their utility on other domains, e.g., indoor PCs.

To the best of our knowledge, no work is dedicated to the interactive exploration of 3DIS results for PC datasets like Scannetv2 and S3DIS in the current literature. As examples of methods that explore PCs, we mention the work of Qin and He [18], who developed a system for analyzing human skeleton detection from PCs aimed at a specific deep learning algorithm. They propose an interesting "segmentation error" view, which we also adopt, but do not provide a general-purpose interface. Zoumpekas and colleagues [19] focused on examining 3D PC part segmentation results, a problem similar to 3DIS, and provided an interface to compare metrics from different models. However, they provide only a single plot for qualitative inspection, a standard 3D scatterplot with categorical features (labels). SOTA 3DIS publications use PC rendering with semantic or instance color mapping to show qualitative results, which are often limited to a small set of scenes [20] [21]. Further, each work uses its tool of choice, which leads to visual incoherence and possible re-work for authors (e.g., compare [11] and [2]).

III. Overview of REIS

We follow the visual analytics design triangle methodology proposed in [22] to guide the construction of REIS. The design triangle requires outlining the three main components of a visual analytics system: the data, the users, and the tasks the users apply to the data. In our case, the **data** are static PCs, which are highly spatial, quantitative, multivariate, and represent a single state of the environment. The **user** can be anyone familiar with the 3DIS problem, like industry engineers and computer vision researchers. Finally, the users' **task** is the evaluation and validation of an instance segmentation model, outlining its positive and negative aspects.

We designed REIS to support a variety of datasets containing PC instance segmentation data. Since each dataset might have distinct representations and instance segmentation models have different ways of outputting results, it can be challenging to integrate them into the same interface. To accomplish this, we propose a decoupled approach, specifying a standard tabular data format (see details on https://github.com/pedro-sidra/pcloud_explorer), and requiring the user to convert their results to this format. We use Python with Plotly Dash for the backend and frontend.

Fig. 1 shows an example of the REIS user interface using the S3DIS dataset [2]. The interface has interactive controls to select and filter variables of interest. The main window shows the rendering of the PC and selected properties. Finally, an Instance Detection Matrix supports selections on a matrix cell to show the corresponding objects in PC rendering.

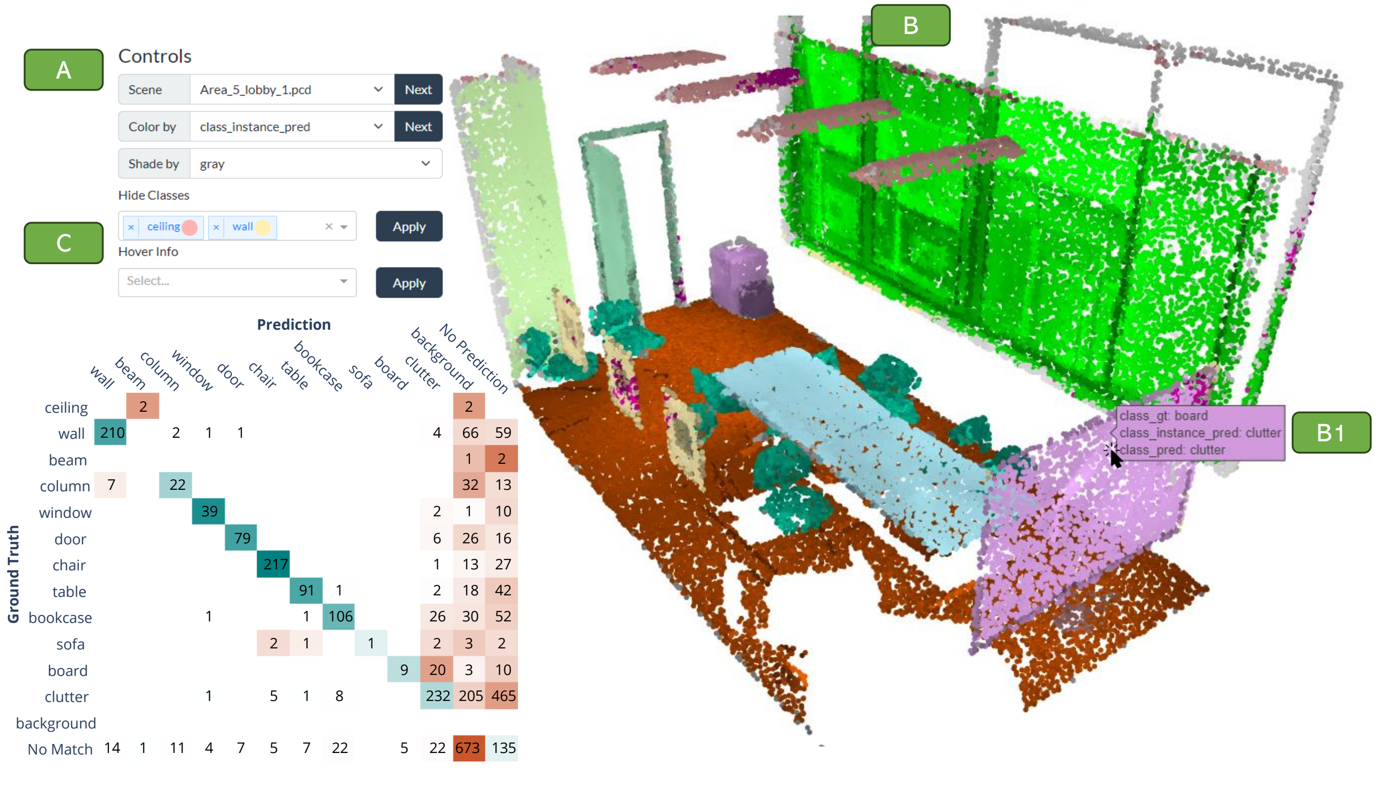
A. Inputting 3DIS Results

The user provides a set of files, a 'load' method, and a list of class names. The input files may be of any format - the 'load' method must interpret them and output a PC in tabular format with the columns of interest. Each column must be given either a numerical or a categorical type and a name, which dictate the type of plot used. The system implements special behavior for certain column names, e.g. 'instance_pred'. Instance-wise attributes should be propagated to the points belonging to the respective instance, which adds redundancy but maintains data-format consistency. The minimum set of columns is the point coordinates (colors optional) plus $\boldsymbol{L}$, $\hat{\boldsymbol{L}}$, $\boldsymbol{I}$ and $\hat{\boldsymbol{I}}$. If $\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}$ are available, the grayscale value $\boldsymbol{gsv} = (\boldsymbol{R} + \boldsymbol{G} + \boldsymbol{B})/3$ is included automatically. Additional columns can be provided and displayed as generic features.

B. User Interface

The user interface is composed of a main window that displays the PC as an interactive 3D scatter plot, using color mappings according to properties the user selects. The system supports discrete color mappings for features with categorical values, like $\boldsymbol{L}$ and $\hat{\boldsymbol{L}}$, and continuous color mappings for features with continuous numerical values like prediction confidences. In this mode, the user can cycle through different files (i.e., scenes) to render the model's inference results. We additionally support using two orthogonal color encodings: *hue* and *luminance*; we call this strategy "shading", and it provides simultaneous rendering of two variables. In addition, we provide an "object prediction" view, which combines $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{L}}$ into a single rendering by using point colors to encode the semantic classifications and bounding boxes to indicate

https://github.com/plotly/dash



Controls

Scene: Area_5_lobby_1.pcd — Next

Color by: class_instance_pred — Next

Shade by: gray

Hide Classes: × ceiling × wall — Apply

Hover Info: Select... — Apply

class_gt: board
class_instance_pred: clutter
class_pred: clutter

	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter	background	No Prediction
ceiling		2										2	
wall	210		2	1	1						4	66	59
beam												1	2
column	7		22									32	13
window				39							2	1	10
door					79						6	26	16
chair						217					1	13	27
table							91	1			2	18	42
bookcase				1			1	106			26	30	52
sofa						2	1		1		2	3	2
board										9	20	3	10
clutter				1		5	1	8			232	205	465
background													
No Match	14	1	11	4	7	5	7	22		5	22	673	135

Fig. 1. REIS rendering user interface. A: controls to select scenes and features of interest. B: interactive PC rendering.; in B1, the "Mouse Hover" feature is visible, showing additional information. C: Instance Detection Matrix; when the user clicks a cell, the corresponding instances are listed in C1.

approximate instance boundaries. We further guide the users' explorations with a compact and comprehensive plot, the *Instance Detection Matrix*, which represents missed instances, false positives, and confusion between classes. The matrix view is interactive, and clicking the cells allows visualizing the corresponding instances to understand each failure case.

C. Shading

REIS provides a "shading" feature that enables simultaneous analysis of two variables in 3D space. The user chooses a discrete variable c_i, which modulates point colors, and a continuous variable q_i, which modulates color intensity. Specifically, c_i determines a base color from a discrete colorscale; this color is then converted into its HSV (Hue, Saturation, Value) components, and the relative value $q_i/(q_{max} - q_{min})$ is used to replace the V component. By default, the system uses $\boldsymbol{gsv}$ as the quantitative variable, since it modulates scene lighting and texture information. Fig. 2 shows an example of this, and Figure 6 shows a different choice of q_i.

One downside of this strategy is that it can lead to high similarity between different discrete colors after applying shading. We build a specific color scale to avoid this problem by exploiting the *hue* channel of the HSV color space:

$$h_k = k\frac{360°}{K+1}, s_k = \begin{cases} 0.5, & \text{if } k \ mod \ 2 = 0 \\ 1, & \text{otherwise} \end{cases}, v_k = 1, \quad (1)$$

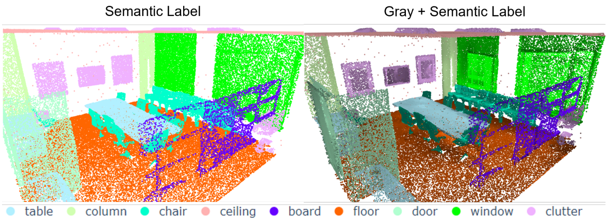


Fig. 2. Comparison of PC rendering using discrete shading and modulation of discrete colors with continuous values (grayscale shading), which improves depth perception in the scene. For example, notice the clear distinction between chair instances and the texture of windows and boards.

where K is the total number of colors in the colorscale, and the chosen color for category k is represented as (h_k, s_k, v_k) in the HSV color space. The saturation channel oscillates between 0.5 and 1 to avoid similarity between colors with similar hues, which tends to happen as K increases. For point cloud 3DIS datasets, K is usually between 10 and 20. Fig. 2 (and all proceeding renderings we show) uses a color scale generated with this strategy using $K = 14$, which is the number of classes on the S3DIS dataset [8] plus an additional `background` class. Although popular datasets usually have a color standard, we argue that the benefit of our shading strategy justifies the change in color scales.

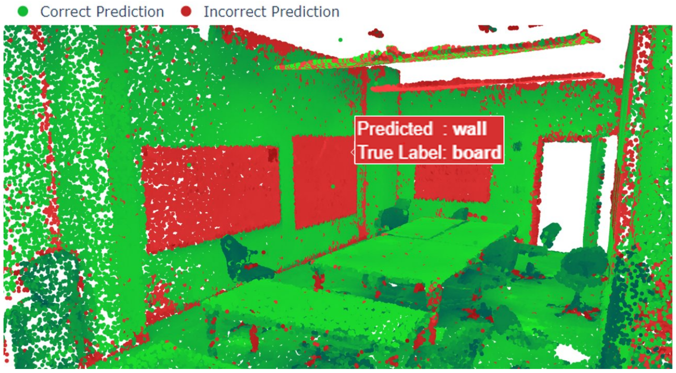


Fig. 3. Rendering model errors on the S3DIS dataset. The box on the center-right of the image shows when the user's mouse hovers over a point.

D. Error Map

A common task in qualitatively analyzing instance segmentation results is to compare model inferences with the Ground Truth (GT). Due to the challenges in PC rendering, this can be an error-prone task, with unnoticed missing or subtle details due to occlusions. We thus provide a rendering view to highlight errors in the semantic predictions. We define the "hit" vector $\boldsymbol{H}$ as follows:

$$\boldsymbol{H} = \begin{cases} True, & \text{if } \hat{L}_i = L_i \\ False, & \text{otherwise} \end{cases}, \tag{2}$$

and map it into green (True) and red (False) colors to indicate the points where the model has mistakes. Through mouse hovering, we detail the mistake by showing the predicted and the actual labels as textual details, and the user can identify model mistakes without comparing two different plots. Fig. 3 shows an example highlighting the mislabeling of windows and boards on the walls, and also a structural column. Notice that the shading strategy with $\boldsymbol{gsv}$ is critical to show scene details in this rendering – without the grayscale/textural information contained in $\boldsymbol{gsv}$, all points with similar colors would be indistinguishable.

E. Instance Detection Matrix

A common way to analyze model performance in classification tasks is to look at the Confusion Matrix (CM). Although not usual, some works on 3D Semantic Segmentation show CMs for point-wise classification [21]. However, the standard concept of a confusion matrix cannot be directly applied in 3DIS. For example, if a predicted instance is not correctly localized (i.e., its IoU is smaller than the acceptance threshold), it cannot be assigned to a GT instance. Such an instance is a false positive, but we cannot assign it to a class-related row in the confusion matrix.

We propose an analog to the confusion matrix that also reflects the instantiation of objects, allowing us to determine errors due to bad instantiation or wrong categorization. We call it *Instance Detection Matrix*, and denote it by $\boldsymbol{\Phi}$. It is a function of a given set of predictions and a given IoU threshold τ, used to determine when an instance is correctly localized, explained next.

We first define a function $mask\_iou(a, b)$, which determines the IoU between the masks $[\boldsymbol{I} = a]$ and $[\hat{\boldsymbol{I}} = b]$. We compute this IoU for all combinations of instance GT ID a and predicted ID b in a given PC, and store all entries with $mask\_iou(a, b) \geq \tau$ in a list. We filter this list to obtain all pairs of "Matched" instances between GT and predictions. First we keep only the highest IoU entry for each unique value of $\boldsymbol{I}$; then we do the same for $\hat{\boldsymbol{I}}$. The resulting entries are the correctly localized instances. Using these entries and their corresponding semantic classifications, we compute the instance-level confusion matrix ϕ like a conventional confusion matrix. We then find $\boldsymbol{FP}$ and $\boldsymbol{FN}$, the false-positive and false-negative vectors, respectively. Both are determined by listing all instance IDs not present in the match list, then taking a histogram of their corresponding semantic classifications. $\boldsymbol{FP}$ reflects predicted instance IDs that are incorrectly localized: they have a predicted category label but no associated GT label; on the other hand, $\boldsymbol{FN}$ accounts for GT instance IDs with no matching prediction: they present a GT category but no corresponding predicted label.

The *Instance Detection Matrix* $\boldsymbol{\Phi}$ is obtained by concatenating the traditional confusion matrix ϕ, which contains instances that are well-localized but wrongly classified, with the source-agnostic instances (called *No Match*) in $\boldsymbol{FP}$ and the target-agnostic instances (called *No Prediction*) $\boldsymbol{FN}$:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi & \boldsymbol{FN} \\ \boldsymbol{FP}^T & 0 \end{bmatrix}. \tag{3}$$

In REIS, $\boldsymbol{\Phi}$ is shown in the main view, next to the PC rendering (Fig. 1). We color the cells in $\boldsymbol{\Phi}$ by the row-normalized value of the cell, with values in the diagonal in green, since they are correct predictions, and other values in red. Besides providing a breakdown of different error types, $\boldsymbol{\Phi}$ can be interacted with by clicking on any cell, which renders the corresponding instances from the original PC. Examples are provided in the next section.

IV. Experimental Results

We demonstrate the functionality of REIS with a case study: investigating the results of the SoftGroup algorithm proposed in [2] on the S3DIS dataset [8]. We used the published code and model from [2], and wrote the required 'load' method to adapt their data format. As done in some instance segmentation approaches, SoftGroup produces an additional category `background` to detect false positives generated during the instance-proposal step. Hence, we append the inference-only class `background` to the original set of 13 classes from S3DIS. Another detail about [2] is that the "*Soft-Grouping*" process might produce more than one instance for the same point. We resolve this (in the same way as the author's visualization implementation) by populating the point cloud with instance IDs in ascending order of instance

We avoid using the Hungarian Algorithm to obtain the optimal matching [11], since it may lead to less interpretable results.

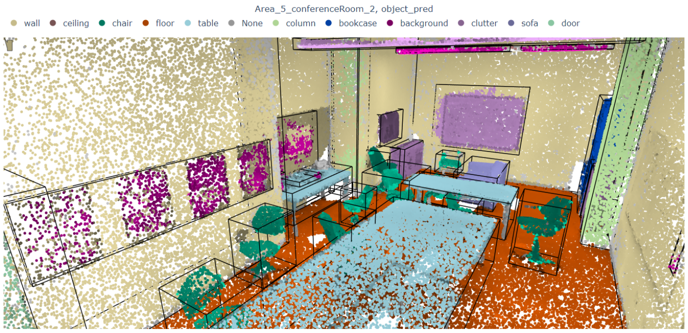


Fig. 4. Object prediction view on Area 5, Conference Room 2 of the S3DIS dataset. Bounding boxes denote approximate predicted object boundaries, and colors denote semantic classification.

classification confidence. This strategy sometimes leads to disjoint or noisy instance predictions since some points from the lower confidence predictions might remain, even if most of the instance is overwritten by a higher confidence one. We keep these artifacts in the visualization and highlight them with bounding boxes in the "Object Prediction" view, as seen in the leftmost instance of `column` and the four instances of `clutter` in Fig. 4.

We guide our investigation using the Instance Detection Matrix $\mathbf{\Phi}$, as shown in the top left of Fig. 5. First, we notice a total of 1,073 entries in the `background` column of $\mathbf{\Phi}$, meaning that several instances were classified into this "artificial category". The count in the last row (673) relates to instances that were not correctly localized, and classifying them as `background` is interesting to avoid false positives. The remaining 400 entries relate to correctly localized instances mislabeled as `background`, from which 205 present `clutter` as the GT label. By clicking on any cell on the `clutter` row, we can visualize the corresponding instances and see that it is a diverse class, which explains the confusion with `background`. However, the remaining 195 entries are related to clearly identifiable categories mislabeled as `background`. This brief visual analysis indicates the pros and cons of using the additional category `background`, putting into context this specific design decision.

Fig. 5 also shows how REIS allows the visualization of the instances shown in matrix Φ. For example, clicking on cell "A" shows instances of `chair` that were not detected, as illustrated in the top-central position of Fig. 5. Along with the individual instances, the system also provides a table discriminating the scenes where each instance was identified, as shown in the bottom left of Fig. 5. This table is also interactive, and clicking on a scene provides its visualization, as illustrated in the bottom right (for the "Area 5, Conference Room 3" scene). We can observe that multiple chairs were merged into a single instance – possibly an error in the instance generator module of Softgroup – that was classified as `background`. In fact, this exact case is mentioned in [11] to highlight their contributions, and it can be quickly discovered using REIS without having to navigate scene by scene exhaustively.

To demonstrate the shading strategy for a continuous variable other than $\boldsymbol{gsv}$, we add a point-wise "Semantic Prediction Confidence" to the point attributes. Fig. 6 shows an example using the predicted labels and confidence. We observe that object boundaries have a lower confidence value, possibly due to the voxel and convolutional nature of the algorithm, which may lead to gradual changes along the boundaries.

V. Conclusions

This paper presented REIS, a visual analytics system designed to explore and analyze instance segmentation results of PCs. Through a combination of rendering techniques, the proposed Instance Detection Matrix, and an interactive interface, REIS offers a domain-specific tool with the potential to streamline qualitative analysis.

Many of the system functionalities aim to present PC information concisely. For example, the shading approach combines one categorical and one continuous variable in a single view, which can be used in many ways to facilitate analysis. Additionally, REIS introduces an error map, which highlights areas where the segmentation model misclassified objects or regions – this avoids a visual comparison of GT and predictions by the user. Bounding boxes also provide additional information, even if less precise than point-wise colors; with them, REIS shows approximate instance information at the same time as semantic labels.

Finally, the system introduces an instance detection matrix $\mathbf{\Phi}$, which expands the concept of a confusion matrix. Besides the inherent information obtained by visual analysis of this matrix, adding interactivity to navigate to the instances responsible for each error case further speeds up visual exploration.

We highlight some cases where REIS helped carry out efficient qualitative analysis, revealing features of the SoftGroup algorithm [2] in the process. We show that its instance refinement classifier seems biased towards the `background` class, a drawback the original authors did not detect or point out. We also visually show the impact of each point possibly corresponding to more than one instance - in that, after collapsing each point to a single instance, some predicted instances might be only partially overridden. Finally, we provide our system and framework to the community. We believe this tool can help speed up and scale up qualitative analysis, empowering the user to go beyond the metrics.

References

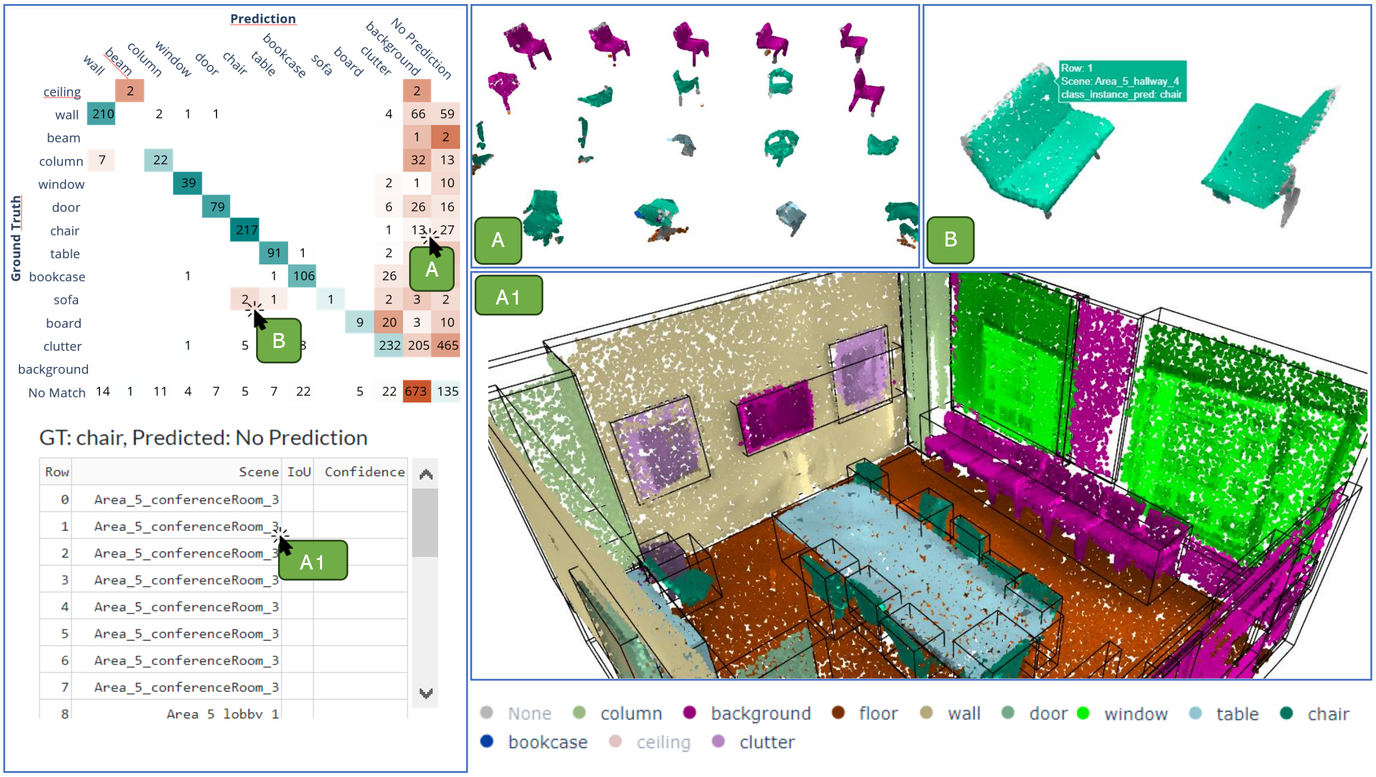
[1] C. Yin, B. Wang, V. J. Gan, M. Wang, and J. C. Cheng, "Automated semantic segmentation of industrial point clouds using respointnet++," *Automation in Construction*, vol. 130, p. 103874, 2021.

[2] T. Vu, K. Kim, T. M. Luu, T. Nguyen, and C. D. Yoo, "Softgroup for 3D instance segmentation on point clouds," in *CVPR*, 2022, pp. 2708–2717.

[3] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on PAMI*, vol. 43, no. 12, pp. 4338–4364, 2020.

[4] T. T. D. Nguyen, H. Rezatofighi, B.-N. Vo, B.-T. Vo, S. Savarese, and I. Reid, "How trustworthy are performance evaluations for basic vision tasks?" *IEEE Transactions on PAMI*, 2022.

[5] R. Jena, L. Zhornyak, N. Doiphode, P. Chaudhari, V. Buch, J. Gee, and J. Shi, "Beyond map: Towards better evaluation of instance segmentation," in *CVPR*, 2023, pp. 11 309–11 318.



Prediction

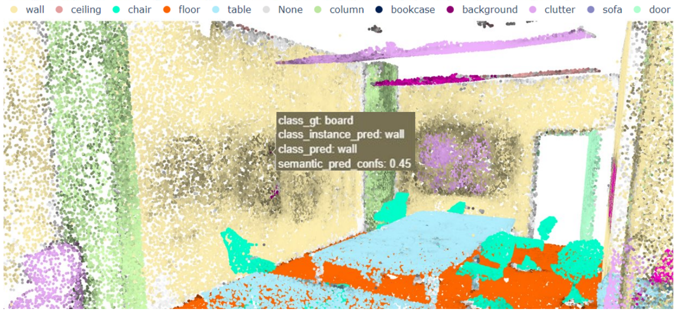
Ground Truth	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter	background	No Prediction
ceiling		2										2	
wall	210		2	1	1						4	66	59
beam												1	2
column	7		22									32	13
window				39							2	1	10
door					79						6	26	16
chair						217					1	13	27
table							91	1			2	[illegible]	[illegible]
bookcase				1			1	106			26	[illegible]	[illegible]
sofa						2	1		1		2	3	2
board										9	20	3	10
clutter				1		5	[illegible]	3			232	205	465
background													
No Match	14	1	11	4	7	5	7	22		5	22	673	135

GT: chair, Predicted: No Prediction

Row	Scene	IoU	Confidence
0	Area_5_conferenceRoom_3		
1	Area_5_conferenceRoom_3		
2	Area_5_conferenceRoom_3		
3	Area_5_conferenceRoom_3		
4	Area_5_conferenceRoom_3		
5	Area_5_conferenceRoom_3		
6	Area_5_conferenceRoom_3		
7	Area_5_conferenceRoom_3		
8	Area_5_lobby_1		

Row: 1
Scene: Area_5_hallway_4
class_instance_pred: chair

Fig. 5. Interactions with the Instance Detection MatrixΦ. In A, the user clicks a cell, and REIS renders the corresponding instances in the PC view; in this case, the user views instances of `chair` that were missed (false negatives). In A1, the user clicks the table to navigate to an instance's corresponding scene to find that many instances of `chair` were missed in Conference Room 3. In B, the user navigates to instances of `sofa` misclassified as `chair`.



class_gt: board
class_instance_pred: wall
class_pred: wall
semantic_pred_confs: 0.45

Fig. 6. Shading strategy to render a continuous variable. Colors represent predicted semantic labels shaded by their corresponding prediction confidences. A hover box (top right) shows the confidence value at the mouse position.

[6] D. Bolya, S. Foley, J. Hays, and J. Hoffman, "Tide: A general toolbox for identifying object detection errors," in *ECCV*, 2020.

[7] A. Xiao, J. Huang, D. Guan, X. Zhang, S. Lu, and L. Shao, "Unsupervised point cloud representation learning with deep neural networks: A survey," *IEEE Transactions on PAMI*, 2023.

[8] 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 *CVPR*, 2016, pp. 1534–1543.

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017, pp. 5828–5839.

[10] M. Chen, Q. Hu, Z. Yu, H. Thomas, A. Feng, Y. Hou, K. McCullough, F. Ren, and L. Soibelman, "Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset," *arXiv preprint arXiv:2203.09065*, 2022.

[11] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask Transformer for 3D Semantic Instance Segmentation," in *ICRA*, 2023.

[12] M. Kolodiazhnyi, D. Rukhovich, A. Vorontsova, and A. Konushin, "Top-down beats bottom-up in 3d instance segmentation," *arXiv preprint arXiv:2302.02871*, 2023.

[13] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *CVPR*, 2020, pp. 4867–4876.

[14] D. Girardeau-Montaut, "Cloudcompare," *France: EDF R&D Telecom ParisTech*, vol. 11, 2016.

[15] M. Schütz *et al.*, "Potree: Rendering large point clouds in web browsers," *Technische Universität Wien, Wiedeń*, 2016.

[16] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[17] B. E. Moore and J. J. Corso, "Fiftyone," *GitHub. Note: https://github.com/voxel51/fiftyone*, 2020.

[18] H. Qin and J. He, "PointCNNVis: A visual analysis system for the interpretability of PointCNN," in *ICCAI*, 2022, pp. 650–656.

[19] T. Zoumpekas, G. Molina, A. Puig, and M. Salamó, "Closed: A dashboard for 3d point cloud segmentation analysis using deep learning." in *VISIGRAPP*, 2022, pp. 403–410.

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017, pp. 652–660.

[21] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *CVPR*, 2019, pp. 3075–3084.

[22] S. Miksch and W. Aigner, "A matter of time: Applying a data–users–tasks design triangle to visual analytics of time-oriented data," *Computers & Graphics*, vol. 38, pp. 286–290, 2014.