

Pushing the limit to near real-time indoor LiDAR-based semantic segmentation

Pierrick Bournez, Jules Salzinger, Marco Cella, Francesco Vultaggio, Francesco d'Apolito, Phillipp Fanta-Jende

AIT, Austrian Institute of Technology - Center for Vision, Automation and Control, Unit Assistive and Autonomous Systems Email: (Jules.Salzinger, Marco.Cella, Francesco.Vultaggio, Francesco.dApolito, Phillipp.Fanta-Jende)@ait.ac.at, pierrick.bournez@student-cs.fr

Keywords: Semantic Segmentation, Transformers, Door and Windows Detection, 3D Building Understanding, UAV, LiDAR

Abstract

Semantic segmentation of indoor 3D point clouds is a critical technology for understanding three dimensional indoor environments, with significant applications in indoor navigation, positioning, and intelligent robotics. While real-time semantic segmentation is already a reality for images, existing classification pipelines for LiDAR point clouds assume a pre-existing map which relies on data collected from accurate but heavy sensors. However, this approach is impractical for high-level task planning and autonomous exploration, which benefits from a rapid 3D structure understanding of the environment. Furthermore, while RGB cameras remain a popular choice in good visibility conditions, such sensors are inefficient in environments where visibility is hindered. Consequently, LiDAR point clouds emerge as a rather reliable source of environmental information in such circumstances. In this paper, we adapt an existing semantic segmentation model, Superpoint Transformer, to LiDAR-based situation where RGB inputs are not available and near real-time processing is attempted. To this end, we simulated our robot's trajectory and leveraged Hidden Point Removal using the open-source dataset S3DIS to train the model. We investigated various strategies such as modifying the interval prediction and thoroughly study its influence on the prediction intervals. Our model demonstrates an improvement from 40 to 67.6 mean Intersection over Union (mIoU) compared to the baseline on simple (floor, ceiling, walls) and complex (doors, windows) classes.

1. Introduction

With the rapid development of 3D sensing technologies, 3D point clouds semantic segmentation have been widely applied in many fields. It plays a crucial role in 3D building reconstruction (Cui et al., 2019), high level task planning (Stache et al., 2023) and robot navigation (Alenzi et al., 2022). Traditionally, point clouds are generated using expensive, heavy, high-precision sensors (Di Stefano et al., 2021), with practitioners focusing on constructing accurate reconstructions without any time constraint. However, when exploring an indoor environment, a mobile vehicle constructs an internal map of the environment. As they have limited energy reserves, they need to understand the environment with lightweight sensors in real-time to make decisions.

This study investigates the performance of deep learning models for near real-time semantic segmentation of structural classes in indoor scenes for a mobile mapping platform. Specifically, we investigate how to train models using only LiDAR data, with points clouds collected by a mobile mapping platform. Our problem considers that the mobile mapping platform maps the environment in real time. Figure 1 shows an example of our model's real time prediction.

Most existing 3D semantic segmentation methods focus on sparse outdoor LiDAR scans for autonomous navigation (Li et al., 2019). However, there is limited research on indoor semantic segmentation with mobile mapping platforms due to a lack of available open-source data. For example, S3DIS (Armeni et al., 2017), a popular indoor dataset, does not provide any sensor trajectory, and the sensor used were primarily RGB sensors.

Our contributions are as follows:

- We adapt the chosen model described in Section 4.1, Superpoint Transformer, to near real-time semantic segmentation of points clouds and speed up the inference pipeline as detailed in Section 4.3. Specifically, we increase the number of neighbors in the preprocessing pipeline to 500 and change the partitioning algorithm to a simple but efficient voxel grid.
- We propose a training procedure in Section 4.2 to train semantic segmentation models on available open-source dataset, here S3DIS, for semantic segmentation in near real-time indoor environments. this training procedure enables the use of datasets that do not provide a sensor trajectory.
- We conduct a comprehensive analysis of the impact of near real-time constraints on a deep learning network performance in this domain. In particular, we analyze the effects of modifying the interval prediction, see Section 5.2.3. We show that incorporating the sensor's field of view during inference into our training scheme, which we introduce through two different methods, is an important factor to improve results.
- We collect a real-world dataset in an office building environment to validate our findings.

2. Related Works

Semantic segmentation of indoor LiDAR mobile mapping point clouds has been mostly addressed using fused RGB features in the 3D robotics community, whereas its LiDAR-only counterpart received comparatively little attention (Alqobali et al., 2023).

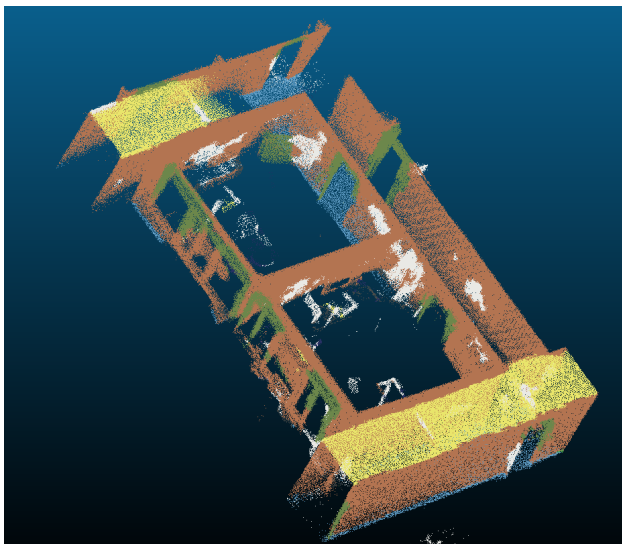


Figure 1. A subset of semantic segmentation that our method produces in near real-time in dataset Office_{10s}.

2.1 Models For 3D Semantic Segmentation

Our focus will be primarily on deep learning (DL) models. DL approaches offer indeed the advantages of reduced handcrafted tuning and increased expressiveness and performance compared to their classical counterparts (Zhang et al., 2018). DL models for semantic segmentation can be classified into three categories: projection-based, point-based, and superpoint-based.

Projection-based methods involve the transformation of 3D point clouds into a 2D format to facilitate the application of established image-based networks (Babacan et al., 2017). This methodology has proven successful, notably in outdoor scenarios (Kong et al., 2023a). This projection requires blind spot sensors. However, in indoor environments, the data obtained lacks these blind spots, reducing the effectiveness of the models.

Point-based networks directly process raw point clouds to infer semantic labels. This paradigm facilitates learning directly from the raw point data and leverages advantages such as efficient sampling techniques and diverse training strategies (Qian et al., 2022). However, these models cannot handle large point clouds and have a limited effective radius (Thomas et al., 2019).

Superpoint-based approaches (Landrieu and Simonovsky, 2018a) is a representation of large 3D point clouds as a collection of interconnected simple shapes called superpoints, in spirit similar to superpixel methods in image segmentation. This partitioning can be achieved by an additional neural network (Hui et al., 2021) or classical computation based on loss outcomes (Robert et al., 2023). While superpoint-based methods entail longer pre-processing times, they allow for feeding the entire point cloud as input to the model at any given time and they can provide an explicit intermediate representation.

2.2 Semantic Segmentation of LiDAR Mobile Mapping Point Clouds

The predominant focus of mobile LiDAR mapping literature has been directed towards outdoor environments.

With the rise of autonomous driving technologies, numerous studies have explored semantic segmentation using mobile mapping LiDAR and image-based point clouds. These studies have

demonstrated successful results in applications such as extracting building (Lu et al., 2014) or predicting structural classes of roads (Alonso et al., 2020). As those point clouds typically exhibit sparsity, outdoor models are specifically designed to tackle this issue (Kong et al., 2023b).

On the other hand, semantic segmentation from indoor mobile mapping systems has received limited focus. Most of the existing literature leverages cameras or RGB-D sensors to use convolutional neural networks (CNNs) or Foundational Models for point cloud classification (Teso-Fz-Betoño et al., 2020). When only 3D Lidar data is available, (Foroughi et al., 2021) projects the data to binary occupancy maps to apply CNNs. Because of the prevalence of cameras in robot navigation (Alqobali et al., 2023), research on LiDAR-only semantic segmentation for indoor mobile mapping remains sparse (Alqobali et al., 2023).

One study utilizes exclusively LiDAR data to discern indoor structural elements for robot navigation (Alenzi et al., 2022). This approach, however, relies on Machine Learning algorithms and has not been trained on publicly available datasets, requiring the manual collection of a custom training dataset. (Cao and Scaioni, 2022) proposed a training procedure to leverage indoor open-source datasets to digitalize external buildings. However, the use case is outdoor and is not designed for mobile mapping point clouds.

3. Problem Definition

We consider a mobile platform operating in an environment equipped with LiDAR sensors that record point clouds. This point clouds is assumed to be mapped with a real-time mapping algorithms. Motivated by (Vultaggio et al., 2023), the mapping algorithm is a LiDAR Simultaneous Localization And Mapping (SLAM) and it is performed in real-time by CT-ICP (Dellenbach et al., 2022). The mobile mapping platform design, a UAV in our case, is outlined in detail in (Bolz et al., 2024)

Our goal is to design a semantic segmentation model that collects mapped point clouds every n seconds and predicts per-point labels.

The model must process and predict mapped point clouds labels within a time $t < n$ to ensure no mapped areas are missed. It should only operate with LiDAR data and no RGB features are provided.

As our focus is on point clouds generated by real-time mapping algorithms, the incoming mapped point clouds may suffer from accumulated drifts from odometry errors and small items may not be accurately recovered using only 3D geometric data. Therefore, we limit our labels and predictions to structural classes within indoor environments, specifically targeting walls, ceilings, floors, doors and windows.

As the mobile mapping platform navigates through the environment, the collected raw point clouds represent the visible surface from a viewpoint. From a training perspective, our objective is to develop a data augmentation method that simulates the visibility of a viewpoint from S3DIS which accurately samples the entire environment area. Mathematically, Given a point p , that corresponds to the center of view, and a point cloud P sampled from surfaces, approximating visibility from a viewpoint is defined as identifying all points $P' \subseteq P$ that would be visible from p if the underlying surfaces were reconstructed. In

the context of point clouds, this question is ill-posed because points cannot occlude one another. Several solutions exist to address this problem, and this concept forms the core motivation for our various training methods described in Section 4.2.

4. Method

This Section presents an overview of our approach. We introduce the deep learning model selected to address this challenge in Section 4.1. Then, we design a training scheme to train the model in Section 4.2. Finally, we detail the modifications to mitigate the absence of RGB features in raw point clouds in Section 4.3. Eventually, we develop our choice and our trade-off to achieve near real time prediction in Section 4.4.

4.1 Superpoint Transformer

To select a semantic segmentation model, we consider several factors. As the end user should define themselves the prediction interval, the model must accommodate small and large point clouds. Additionally, we propose a training procedure that requires an explicit intermediate representation of the data to approximate visibility in large point clouds (see Section 4.2). Consequently, the model has to include this intermediate representation. Therefore we employ Superpoint Transformer (Robert et al., 2023), a model that meets all these requirements. It is designed to handle large and small point clouds by computing an intermediate representation. The author claims that the model has faster training and inference times compared to other available models. When RGB features are provided, it demonstrates good performance on S3DIS with 76.0 mIoU for 6-fold validation (Robert et al., 2023).

This Section outlines the components of Superpoints Transformer, readers are referred to the original article for more details. The models can be separated into four keys components:

- **Feature Construction:** The model uses k-nearest neighbors (kNN) to obtain neighborhoods for each point and subsequently compute handcrafted features. In addition to RGB features (which are absent in our use case), PCA-based features like linearity, planarity, scattering (Demantké et al., 2012), and verticality (Guinard and Landrieu, 2017) are used. A feature called elevation, which is defined as the distance between a point and the ground below is determined with RANSAC. In Section 4.3, we explain how we modify this original list of features.
- **Partitioning the Point Clouds:** To process large areas, the partition-based models first compute a superpoint representation of the point clouds. The core assumption is that superpoints, being a geometrically coherent cluster of points, will also constitute a good input to predict a semantically coherent set. To compute these superpoints, the features are viewed as a signal defined on points. This signal is then approximated as a piece-wise constant function by solving an energy minimization problem. The resulting constant areas form partitions P_i of the initial point cloud, which constitute the new point cloud. The process is iterated to obtain different levels of partition. The resulting hierarchical partitions form several point clouds that will be the nodes of the graph given as input to the model. These different levels of partitioning provide information at various levels of detail. The partitioning algorithm used originally is Parallel Cut-Pursuit (Raguet and Landrieu,

2019). In Section 4.4.2, we replace this algorithm by a voxel grid.

- **Graph Construction:** Once the partitions are constructed, edges are created between superpoints to build a *superpoint-graph*. The goal is to create edges only between close superpoints. Specifically, edges are created between superpoints in P_i if their closest points are within a gap distance ϵ_i . This constructed graph is the actual input of the network.
- **Semantic Segmentation with Transformers** Finally, a U-net-like model with self-attention layers predicts the semantic labels of the hierarchical partition based on the computed graph. In this paper, we do not modify the model's architecture and refer readers to the original paper for an exact definition.

4.2 Training Procedure

When training with S3DIS, most authors feed complete point clouds (Qian et al., 2022) or spherical sections (Thomas et al., 2019) to the model. However, this is unrealistic due to the model having access to occluded areas. Therefore, we structured our training process in two stages, a pretraining phase and different possible fine-tuning phases. This prevents the model from relying on data that wouldn't be visible in real scenario and simulates mobile mapping patterns. An overall summary of our training pipeline can be found in Figure 3

4.2.1 S3DIS we first pretrain on S3DIS without adding any new data augmentation. Training with the dataset alone proves insufficient in practice. For example, it fails to properly detect doors and windows (see Table 5 for detailed results)

4.2.2 Hidden point removal (HPR) As S3DIS provides entire office areas, it lacks the patterns that would typically come from a mobile mapping platform. To address this, we can pre-train or fine-tune the model with additional data augmentation using a HPR algorithm (Katz et al., 2007). To compute each input mapped point cloud at training time, a random point in the point clouds is chosen as the center for the algorithm. This randomness simulates different viewpoints and helps the model generalize better by exposing it to various perspectives and occlusion scenarios. As the algorithm computes convex hulls, it has high computational complexity. Therefore, we applied it to the superpoints themselves. This data augmentation is advantageous because it uses the raw data and doesn't require any additional information or simulation.

4.2.3 Simulating trajectory: S3DIS_{sim} S3DIS provides complete meshes generated from data collected from a static camera system. Given that meshes are available in public datasets and a digital twin of our robot and sensor (Vultaggio et al., 2023) is provided, we simulate the robot's trajectory in Gazebo (Koenig and Howard, 2004) within the mesh and trained on the collected sensor data. The collected sensor data is after split in n -seconds intervals to form S3DIS_{10s}. This approach more accurately mimics our sensor's patterns, see Section 5.1.2 for a complete description of our data collection.

4.3 Adapting the Model to RGB Absence and Noisier Point Clouds

Most prior studies, including SuperPoint Transformer, use RGB as additional input features in the point cloud. RGB features are

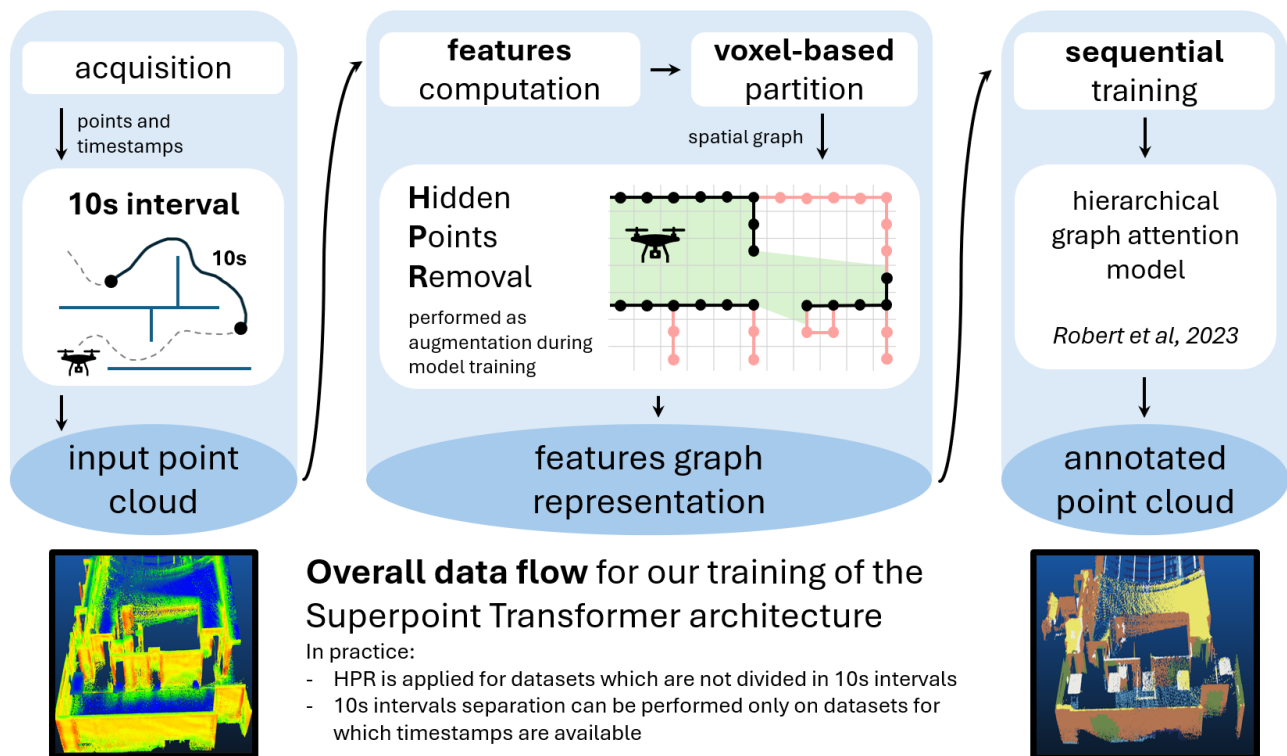


Figure 2. Schematic representation of our method. The input to the model is a graph constructed from the sparse point cloud.

directly acquired by the sensors and are not affected by numerical instability or subjective handcrafted decisions. Therefore, they constitute a very valuable source of information for the partition construction. To compensate for its absence, we evaluate additional features and select those that enhance our ability to differentiate doors and windows from walls and ceilings. Specifically, we decide to add anisotropy (Hackel et al., 2016) which is defined as $(\lambda_1 - \lambda_3)/\lambda_1$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the eigenvalue of the covariance tensors computed from the spatial distribution of points within each superpoint.

Furthermore, the elevation feature is computed using RANSAC, which often fails with a small number of points. Since our preprocessing pipeline on S3DIS_{10s} often results in a small number of points, we decide to remove the elevation feature. Based on a hyperparameter search, we remove the surface feature defined in (Landrieu and Simonovsky, 2018b) and we stabilize the computation of the features by increasing the number of neighbors considered from 50 to 500.

4.4 Pushing to Near Real-time Inference

In this section, we outline the methods and choices implemented to achieve near real-time inference with Superpoint Transformers.

4.4.1 Interval Prediction Selection Assuming a SLAM algorithm is already in place and provides a stream of mapped point cloud data, we can perform inference within various time windows, albeit with increased noise for shorter intervals.

As the prediction interval decreases, we might expect a decreased mIOU in the semantic segmentation analysis. We opted to predict every $n = 10$ seconds, as this interval produced results comparable to those obtained by processing the entire map, see Figure 9. We provide an extensive analysis of the size of time window in the result Section 5.2.3

| Operation [s] | Our pipeline | Initial baseline |
|--------------------------------|--------------|------------------|
| I.O reading time | 0.05 ± 0.1 | 0.05 ± 0.1 |
| Feature Construction | 1.5 ± 0.7 | 0.2 ± 0.007 |
| Parallel Cut Pursuit algorithm | . | 1.0 ± 0.2 |
| Grid Voxel algorithm | 0.01 ± 0.001 | . |
| Graph Construction | 0.5 ± 0.1 | 0.6 ± 0.1 |
| model inference | 0.05 ± 0.01 | 0.06 ± 0.01 |
| Total | 5.8 ± 1.4 | 4.9 ± 1.1 |

Table 1. Comparison of preprocessing performance for each major step using our optimized pipeline on a 10-second point cloud interval. Results are in seconds, with "." indicating the absence of the operation in the respective pipeline.

4.4.2 Computation Trade-off In our use case, we aim to make predictions on mapped point clouds in near real-time. Although the authors of Superpoint Transformers (Robert et al., 2023) claim that the inference time is close to 2 seconds, preprocessing can take on average up to 20 seconds for 10-second intervals on inference with our setup.

The Cut-Pursuit partition algorithm can occupy up to 40% of the total preprocessing time. Given that the quality of the current partitioning does not appear to be a limiting factor for model accuracy, we replace Parallel Cut Pursuit with a voxel grid partition during training on 10-second intervals. This partition algorithm is faster than Parallel Cut Pursuit, with an average speed of 0.1 seconds compared to 1 second for 10-second intervals (see Table 1).

This change from PCP to grid voxel can only be made when training on S3DIS_{10s}, as grid voxel partitioning cannot fit into GPU memory for really large point clouds like S3DIS and requires a voxel size that must be manually defined for each dataset.

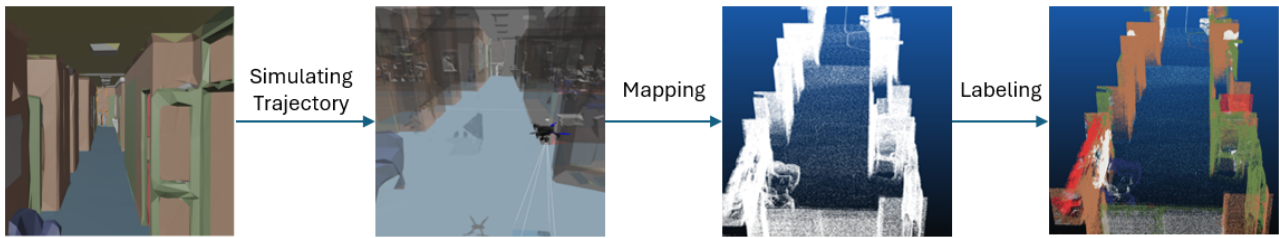


Figure 3. Our Simulation workflow to acquire $S3DIS_{sim}$ from $S3DIS$ begins by creating waypoints within the mesh to simulate our mobile mapping platform. Next, we map the incoming point clouds using a SLAM method and align the output point clouds with the initial mesh to obtain the labeled point cloud.

| Dataset name | Short description | Train set | Val/Test set |
|----------------|---|------------|--------------|
| $S3DIS$ | Original $S3DIS$ dataset | Area 1-4,6 | Area 5 |
| $S3DIS_{sim}$ | Simulated mesh of $S3DIS$ | Area 1-4,6 | Area 5 |
| $S3DIS_{10s}$ | Points clouds collected every $n = 10$ seconds from $S3DIS_{sim}$ | Area 1-4,6 | Area 5 |
| $Office_{10s}$ | Real office mapped with our sensors. points clouds are collected every 10 seconds | | All data |

Table 2. Summary of all datasets used and their corresponding use in training, validation and test set

5. Experiments and Results

5.1 Datasets

A summary of all our development datasets can be found in table 2.

5.1.1 $S3DIS$ $S3DIS$ (Armeni et al., 2017) is an RGB-D dataset that consists of 6 indoor areas. It is one of the most widely used indoor dataset in the literature, meshes and point clouds are provided. We discarded the RGB features.

5.1.2 $S3DIS_{sim}$ and $S3DIS_{10s}$ We simulate our robot’s trajectory in $S3DIS$ with Gazebo. As $S3DIS$ ’s mesh contains holes, using automatic exploration algorithms to simulate data collection is challenging. Instead, the drone’s path is executed using manually-defined waypoints. SLAM is performed based on the UAV’s trajectories, and - to correct for any accumulated drift - the point cloud is annotated by aligning the mesh with the resulting point cloud. The entire simulated mapped point clouds will be called $S3DIS_{sim}$. This point cloud is then further split into n -second intervals, which serve as the model’s inputs. We will denote this new data as $S3DIS_{10s}$. An overall workflow of the simulation can be found in Figure 3.

5.1.3 $Office_{10s}$ Using our mobile mapping platform (Bolz et al., 2024), we map a working office which consists of a building with two floors, one hall, and 16 rooms. This data set reflects a true use case. We manually label the resulting point cloud. Small items such as table and chair exhibit significant noise due to the sensor’s noise and SLAM inaccuracies.

We notice strong reflections from windows that mimic the room where the platform is located, see Figure 4. Labeling these reflected values is prone to human interpretation, so we decide to categorize them as clutter. Since reflectivity is challenging to simulate in a virtual environment and is absent from our photogrammetric datasets, we report the mIOU both with and without it. The mIOU, when accounting for reflectivity, is significantly lower because reflected points may have distinctive shapes, like walls or windows, that the models will incorrectly predict instead of the chosen "clutter" class.

5.2 Results Analysis

We present our comprehensive results in Table 3. The most noticeable behavior is a large gap between our results on real data

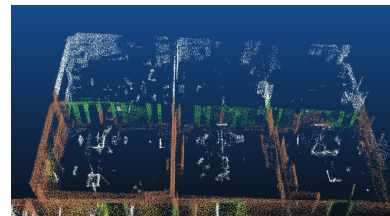


Figure 4. Windows are colored in green. reflection are labeled as clutter (white).

| Training Procedure | $Office_{10s}$ | $Office_{10s}$ no reflection | $S3DIS_{10s}$ | $S3DIS$ |
|-----------------------------------|----------------|---------------------------------|---------------|-------------|
| $S3DIS$ (baseline) | 31.6* | 35.3* | 40.0* | 63.9 |
| $S3DIS$ | 32.5 | 37.9 | 41.5 | 62.2 |
| $S3DIS_{sim}$ | 32.8 | 39.8 | 58.5 | 70.0 |
| HPR | 35.0 | 42.4 | 59.8 | 68.2 |
| $S3DIS_{sim}$ +HPR | 34.1 | 40.8 | 52.7 | 69.2 |
| $S3DIS_{10s}$ | 37.1 | 41.9 | 64.1 | 69.4 |
| $S3DIS_{sim}$ + $S3DIS_{10s}$ | 38.0 | 42.2 | 65.7 | 69.7 |
| $S3DIS_{sim}$ +HPR+ $S3DIS_{10s}$ | 37.1 | 42.8 | 67.6 | 68.2 |

Table 3. Main results table. * indicates the removal of the feature elevation during training and preprocessing because its computation on sparse clouds is unsuccessful. + denotes sequential training. The baseline contains no changes from (Robert et al., 2023) besides those indicated for compatibility reasons.

($Office_{10s}$) and simulated data ($S3DIS_{10s}$). We analyze this gap in Section 5.2.1. Then, we investigate the effectiveness of our training modifications in Section 5.2.2. Finally, we analyze the impact of modifying the prediction interval in Section 5.2.3.

5.2.1 The gap between real and simulated data Overall, we observe a significant mIOU discrepancy between testing on $S3DIS$ (70.0 at best) and on $Office_{10s}$ (42.8 at best), see Table 3. We identify several contributing factors to partially explain this mIOU drop:

- wall sections are sometimes mislabeled as doors, especially in $Office_{10s}$ which contains long sections of wall (Figure 5): our training datasets contain closed doors which are indistinguishable from walls without RGB features;
- our predictions become less accurate as the distance from the drone increases (Figure 6): our training datasets con-







| Color | Label | Color | Label |
|---|---------|---|---------|
|  | Ceiling |  | Floor |
|  | Door |  | Window |
|  | Wall |  | Clutter |

Table 4. Colormaps

tain mainly small rooms and few long corridors, contrary to Office_{10s};

- reflections in Office_{10s} (Figure 4) are rarely correctly segmented: they do not appear in our training datasets;
- doors and windows are harder to tell apart (Figure 7): the drone flies relatively high, so only the upper parts of items are visible in Office_{10s}.



Figure 5. Example of a wall mislabeled as a door due to labeled mesh inaccuracies in Office_{10s}.

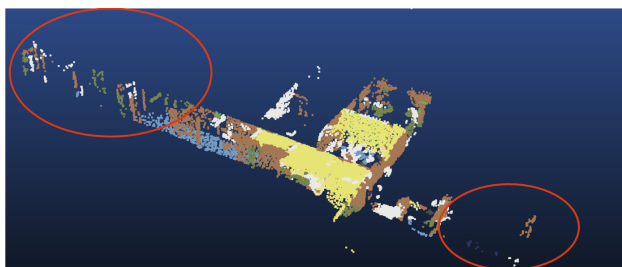


Figure 6. Predictions become less coherent as the point cloud is farther from the drone's position in Office_{10s}.

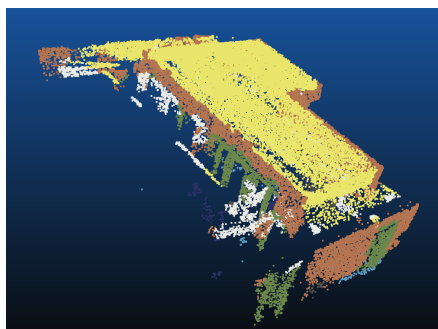


Figure 7. Example of good prediction in Office_{10s}.

It can be noted that the mIoU on S3DIS without RGB features decrease from 65.0, as reported in (Robert et al., 2023) to 63.9 in our study. However, RGB features were omitted only during the training phase and still utilized during the partitioning computation in (Robert et al., 2023), which stabilizes the features and the partitions.

| Method | mIoU | ceiling | floor | wall | window | door | clutter |
|--|-----------|-------------|-------------|-------------|-------------|-------------|-------------|
| S3DIS | 32.5 | 64.9 | 32.6 | 57.4 | 0.8 | 9.1 | 19.0 |
| S3DIS _{sim} +S3DIS _{10s} | 38 | 65.4 | 44.9 | 58.3 | 12.4 | 16.0 | 21.0 |
| S3DIS _{sim} +HPR+S3DIS _{10s} | 37.1 | 65.2 | 42.5 | 55.4 | 14.1 | 13.4 | 21.2 |

Table 5. Per-class results on Office_{10s} (with reflections)

5.2.2 Training modification

Noise mitigation measures: the mIoU marginally decreases from 63.9 to 62.2 on S3DIS but marginally increases for all other datasets, for example going from 31.6 to 32.5 on Office_{10s}. This seems to confirm that the changes from Section 4.3 are targeted at noisy and low-resource scenarios. As S3DIS is a very precise reconstruction, increasing the number of neighbors in feature computation results in lower mIoU, which can be ascribed to a more averaged out feature representation. However, we achieve better results with noisier SLAM point clouds and in mobile scenarios because we effectively mitigate the noise issues.

Training on simulated data: training on S3DIS_{sim} instead of S3DIS yields marginal improvements on the Office dataset and large improvements on the S3DIS datasets. In particular, the mIoU on S3DIS_{10s} goes from 41.5 to 58.5. Training on S3DIS_{sim} seems more effective for mobile scenarios. This can be attributed to a smaller distribution gap between the training and test data.

Hidden Points Removal: When using HPR, and thus introducing the notion of visibility which lacks in S3DIS_{sim}, the mIoU on Office_{10s} increases from 32.8 to 35.0. This is only three points lower than using S3DIS_{10s} for training but is a much simpler method to perform in general, making it a considerable option despite Office_{10s}'s higher fidelity. In general, we notice instability when training with HPR. Pretraining with S3DIS_{sim} does not seem to solve the issue, even resulting in an mIoU decrease of at least one point on every dataset. Decreasing this instability might be a valid research direction to further improve the results in the future.

Pretraining with S3DIS_{sim}: Pre-training on S3DIS_{sim} yields mediocre results which are inconsistent across our various datasets, and does not seem to facilitate convergence in general.

Training on S3DIS_{10s}: The mIoU increased from 32.8 on Office_{10s} to up to 38.0 when training with S3DIS_{10s}. This indicates that the influence of training on clouds collected during a short interval is significant. This constitutes our main result and supports our claim that incorporating the notion of visibility is essential when training a model for online use.

Best models: The two most effective models are those trained sequentially, first on S3DIS_{sim} (although as pointed out earlier this step's contribution might be minimal), followed by S3DIS_{10s}. One of them includes HPR, the other not. A more detailed analysis of the two optimal models is provided in Table 5. As anticipated, the model demonstrates high accuracy for simpler classes such as ceilings and walls. Both models differ in their accuracy for walls and windows, which are already challenging in S3DIS (Robert et al., 2023). The S3DIS_{sim}+S3DIS_{10s} model has a better mIoU for doors, 16.0 over 13.4, compared to the S3DIS_{sim}+HPR+S3DIS_{10s} model, which has an mIoU of 14.1 over 12.4.

Furthermore, the floor mIoU increases from 32.6 to 44.9 and 42.5 respectively. Our sensor, being placed on top of the UAV,

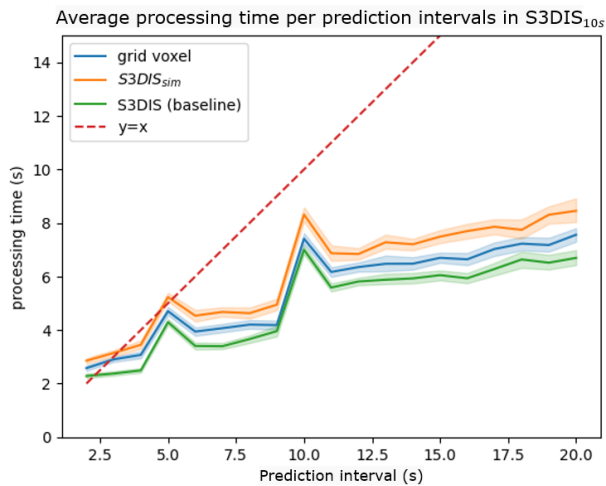


Figure 8. processing time for different pipelines

does not capture the floor with a high point density, which makes it more challenging than terrestrial setups. Our methods do however succeed in significantly improving the results for this class.

5.2.3 Selecting the prediction interval Detailed graphs of the mIOU and processing time as a function of the prediction interval can be seen in Figures 8 and 9. The prediction interval should satisfy two criteria: it should maintain an mIOU comparable to that achieved with the entire dataset, and the processing time has to be shorter than the prediction time (ie., if we predict every 10 seconds, the inference should take less than 10 seconds). Above 6 seconds, our pipeline can process data faster than the incoming point clouds, satisfying the second condition. As shown in Figure 9, mIOU stabilizes after prediction intervals of 10 to 12 seconds (although it continues to increase). Given these two factors, we recommend choosing an interval of at least 10 seconds, and we provide the results on 10 seconds as the most challenging case. Interestingly, the model still benefits from inference on the full scene even when trained only on fixed prediction intervals of 10 seconds.

The preprocessing benchmark is executed on an Intel(R) Xeon(R) Silver 4116 CPU for a total of 48 threads running at 2.1 GHz, and two GeForce RTX 3090 GPUs, each with 24 GiB of VRAM. Despite numerous performance improvements, our changes make the pipeline slightly slower compared to the initial pipeline proposed by (Robert et al., 2023). This is mostly due by the increased number of neighbors in the feature computation, which is required for consistency under increased noise. However we remain competitive with respect to the initial model while having a better mIOU on our Office_{10s} dataset.

6. Conclusion

This paper presents a comprehensive analysis of indoor semantic segmentation in near real-time. Specifically, from a model perspective, we examine how to train such models using available datasets and how to adapt them to this modality. From a use case perspective, we optimize the model to operate more efficiently and evaluate the impact of time constraints on model performance.

We specifically study the case where the LiDAR has a 360° FoV. Our data augmentations and, especially, Hidden Point Removal, reflect this perspective. If the FoV of the sensor were

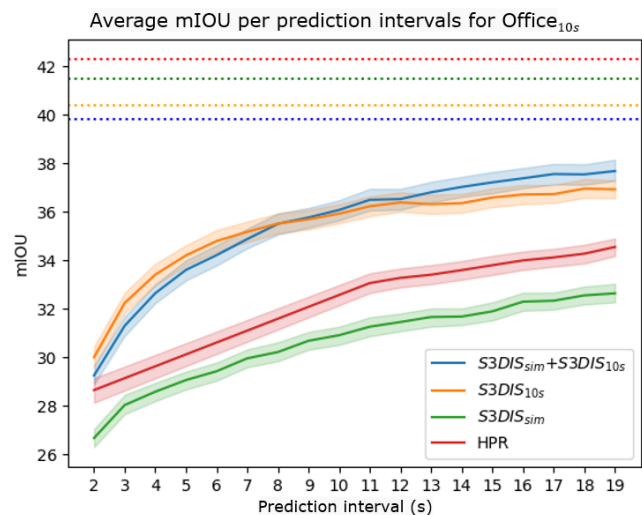


Figure 9. mIOU per interval time for three trained models. the dot line are the reported mIOU on the whole office with reflections.

different, the data augmentations or model modifications would need to be adjusted accordingly.

In future work, we aim to integrate loop closure with the mapping algorithm to enhance the robot's guidance. Identifying doors and windows could help the model detect new, promising free spaces. Additionally, we intend to explore whether expanding the training data could further improve the model's performance.

7. Acknowledgments



Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or DG DEFIS. Neither the European Union nor the granting authority can be held responsible for them.

References

- Alenzi, Z., Alenzi, E., Alqasir, M., Alruwaili, M., Alhmiedat, T., Alia, O. M., 2022. A semantic classification approach for indoor robot navigation. *Electronics*, 11(13), 2063.
- Alonso, I., Riazuelo, L., Montesano, L., Murillo, A. C., 2020. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *IEEE Robotics and Automation Letters*, 5(4), 5432–5439.
- Alqobali, R., Alshmrani, M., Alnasser, R., Rashidi, A., Alhmiedat, T., Alia, O. M., 2023. A Survey on Robot Semantic Navigation Systems for Indoor Environments. *Applied Sciences*, 14(1), 89.
- Armeni, I., Sax, A., Zamir, A. R., Savarese, S., 2017. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*.

- Babacan, K., Chen, L., Sohn, G., 2017. Semantic segmentation of indoor point clouds using convolutional neural network. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, 101–108.
- Bolz, W., Cella, M., Maikisch, N., Valtaggio, F., D’Apolito, F., Bruckmüller, F., Sulzbachner, C., Fanta-Jende, P., 2024. A robust lidar-based indoor exploration framework for uavs in uncooperative environments. *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 168–176.
- Cao, Y., Scaioni, M., 2022. A pre-training method for 3d building point cloud semantic segmentation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2, 219–226.
- Cui, Y., Li, Q., Yang, B., Xiao, W., Chen, C., Dong, Z., 2019. Automatic 3-D Reconstruction of Indoor Environment With Mobile Laser Scanning Point Clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(8), 3117–3130.
- Dellenbach, P., Deschaud, J.-E., Jacquet, B., Goulette, F., 2022. Ct-icp: Real-time elastic lidar odometry with loop closure. *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 5580–5586.
- Demantké, J., Mallet, C., David, N., Vallet, B., 2012. Dimensionality based scale selection in 3D lidar point clouds. *The international archives of the photogrammetry, remote sensing and spatial information sciences*, 38, 97–102.
- Di Stefano, F., Chiappini, S., Gorreja, A., Balestra, M., Pierdicca, R., 2021. Mobile 3D scan LiDAR: A literature review. *Geomatics, natural hazards and risk*, 12(1), 2387–2429.
- Foroughi, F., Wang, J., Nemat, A., Chen, Z., Pei, H., 2021. Mapsegnet: A fully automated model based on the encoder-decoder architecture for indoor map segmentation. *IEEE Access*, 9, 101530–101542.
- Guinard, S., Landrieu, L., 2017. Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42, 151–157.
- Hackel, T., Wegner, J. D., Schindler, K., 2016. Contour detection in unstructured 3d point clouds. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1610–1618.
- Hui, L., Yuan, J., Cheng, M., Xie, J., Zhang, X., Yang, J., 2021. Superpoint network for point cloud oversegmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5510–5519.
- Katz, S., Tal, A., Basri, R., 2007. Direct visibility of point sets. *ACM SIGGRAPH 2007 papers*, 24–es.
- Koenig, N., Howard, A., 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, 3, Ieee, 2149–2154.
- Kong, L., Liu, Y., Chen, R., Ma, Y., Zhu, X., Li, Y., Hou, Y., Qiao, Y., Liu, Z., 2023a. Rethinking range view representation for lidar segmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 228–240.
- Kong, L., Ren, J., Pan, L., Liu, Z., 2023b. Lasermix for semi-supervised lidar semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21705–21715.
- Landrieu, L., Simonovsky, M., 2018a. Large-scale point cloud semantic segmentation with superpoint graphs. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4558–4567.
- Landrieu, L., Simonovsky, M., 2018b. Large-scale point cloud semantic segmentation with superpoint graphs. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4558–4567.
- Li, X., Du, S., Li, G., Li, H., 2019. Integrate point-cloud segmentation with 3D LiDAR scan-matching for mobile robot localization and mapping. *Sensors*, 20(1), 237.
- Lu, Z., Im, J., Rhee, J., Hodgson, M., 2014. Building type classification using spatial and landscape attributes derived from LiDAR remote sensing data. *Landscape and Urban Planning*, 130, 134–148.
- Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B., 2022. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35, 23192–23204.
- Raguet, H., Landrieu, L., 2019. Parallel cut pursuit for minimization of the graph total variation. *arXiv preprint arXiv:1905.02316*.
- Robert, D., Raguet, H., Landrieu, L., 2023. Efficient 3d semantic segmentation with superpoint transformer. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 17195–17204.
- Stache, F., Westheider, J., Magistri, F., Stachniss, C., Popović, M., 2023. Adaptive path planning for UAVs for multi-resolution semantic segmentation. *Robotics and Autonomous Systems*, 159, 104288.
- Teso-Fz-Betoño, D., Zulueta, E., Sánchez-Chica, A., Fernandez-Gamiz, U., Saenz-Aguirre, A., 2020. Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot. *Mathematics*, 8(5), 855.
- Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE/CVF international conference on computer vision*, 6411–6420.
- Valtaggio, F., d’Apolito, F., Sulzbachner, C., Fanta-Jende, P., 2023. Simulation of low-cost MEMS-LiDAR and analysis of its effect on the performance of state-of-the-art slams. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48, 539–545.
- Zhang, R., Li, G., Li, M., Wang, L., 2018. Fusion of images and point clouds for the semantic segmentation of large-scale 3D scenes based on deep learning. *ISPRS journal of photogrammetry and remote sensing*, 143, 85–96.