6

Vision-language Embeddings in Large Scale LiDAR SLAM for Terrain Segmentation

Pēteris Račinskis, Janis Arents, and Modris Greitans

Institute of Electronics and Computer Science (EDI), Latvia

Abstract

This paper aims to showcase the potential of extending the open-set semantic approach enabled by vision-language model embeddings for autonomous robot perception through the introduction of a vector-valued voxel mapping software package - SLAMVDB, short for "SLAM vector database", capable of terrain segmentation after introducing a lightweight closed-set classifier in the map retrieval stage. This technology integrates point clouds and camera images to construct semantic maps, with a focus on applications in outdoor autonomous robotics. A single neural network inference pass on the image generates pixel-wise embeddings, which are fused into an octree-based spatial data structure. Information in the map is then retrieved by performing vector similarity searches with respect to text or image embeddings, or specific lookup vectors obtained through supervised learning on a discrete class set in the terrain segmentation task. The map is shown to be capable of performing terrain segmentation for robot navigation, with overall accuracy as high as 87.35% on track *00000* of the Rellis-3D data set, with a coarse-grained 5-class ontology. This paper describes the implementation, verification procedure and evaluation results of the system.

Keywords: open-set semantics, vision-language models, edge AI, SLAM, robotics.

6.1 Introduction and Background

In outdoor-focused industries such as agrifood and forestry, demographic shifts as well as the often physically demanding, tedious and even dangerous nature of the work present the looming prospect of labor shortages [1]. While current staffing levels may still be sufficient to maintain productivity, careers in these industries are failing to attract sufficient numbers of replacements as the current workforce retires, resulting in an "aging out" phenomenon. Other industries, such as manufacturing, have for decades successfully employed robotics to both increase productivity and combat labor shortages. In predictable environments the go-to method for deploying robotic systems remains programming by hand. Advances in machine learning, especially in terms of perception, have enabled robots to adapt to their surroundings through object detection and pose estimation [2]. Moreover, when combined with mobile bases, mass adoption of robotic solutions has also reached beyond the static manufacturing line into domains such as warehouse management [3]. However, it must be noted that technology readiness and adoption rapidly decline as the operating environments trend towards the less structured – such as semi-structured farms [4] and unstructured forests.

As soon as a robot departs the confines of a predefined environment, autonomy requires tackling the challenge of Simultaneous Localization and Mapping (SLAM) [5]. However, for a robot to perform complex tasks, its environmental models must be constructed to be not only spatially accurate but also imbued with semantic annotations that encode the meaning and function of objects and surfaces. Prior methodologies in semantic mapping have typically followed a discrete classification paradigm. The discrete semantic mapping approach is exemplified by systems such as *SemanticFusion* [6], which produces surfel cloud maps with class tags. The introduction of Vision-Language embedding Models (VLMs), such as CLIP [7], has paved the way for a new, more flexible approach: open-set semantics. Through self-supervised joint learning of text and image embedding models on large-scale internet-sourced datasets, these models learn to project images and their descriptions to nearby points in a high-dimensional latent vector space. Such a capability enables zero-shot generalization to novel concepts without requiring retraining of the entire vision model. Before this technology can be effectively leveraged in robot control, multiple hurdles need to be overcome. The first involves taking the step from embedding images as a whole, as in the original CLIP model, to producing a latent space embedding for each pixel – resulting in a *feature map*. One approach, as taken by *LSeg* [8], performs

end-to-end training using a pre-trained text embedding model. An image segmentation model is directly trained to produce a feature map, using vector embeddings of class names for the final vector similarity comparison. A more involved, two-stage route is taken by *conceptfusion* [9]. They use the Segment Anything Model (SAM) [10] to first partition the image into proposed masks, then individually crop a region around each mask, embed it as a single vector, and add this resulting embedding vector to the feature value of each pixel in the object mask.

With a feature map in hand, the next step is volumetric data association. For example, c*onceptfusion* adopts the same surfel approach as taken by the previously described discrete-valued *SemanticFusion* system, producing an unsorted list of surfel elements. Alternatively, one may also construct implicit spatial representations by modifying Neural Radiance Fields (NeRFs) to predict the latent space embedding of each spatial coordinate rather than its appearance [11]. In this case, the "map" is not stored as an explicit data structure, rather being encoded in the weights of the radiance field model. However, one of the classic approaches to building volumetric maps containing arbitrary data is the octree grid map. Indeed, this is also the approach adopted in the authors' previous work [12], where the *OctoMap* [13] software package was extended to permit storing latent space embeddings and allow vector similarity search operations – yielding the *Vectree* system.

The principal goal of the work described in this chapter was demonstrating the applicability of open-set semantics for a different use-case and different information retrieval pattern. The software package introduced here, SLAMVDB [14], is first and foremost designed to accomplish a critical task for autonomous, mobile robotics: terrain segmentation for navigational purposes. Existing approaches to terrain segmentation often rely on models trained specifically for this task on limited datasets, which may not generalize well to novel environments, a challenge addressed by architectures like GANav [15]. This, while achieving substantial improvements over the then-SotA, requires a highly bespoke vision model architecture. The novel contribution outlined here is to instead leverage the open-vocabulary power of VLMs, applying an off-the-shelf, pre-trained *LSeg* vision model with frozen weights to the terrain segmentation problem. We use supervised learning only for creating a specialized set of query vectors, optimized on a small data set to build a discrete classifier operating on latent space embedding features, which can perform retrieval operations in either the image or voxel domain – the latter allowing many observations of a voxel to be fused and averaged over time.

6.2 Approach and Implementation

Before the problem of semantic mapping can be considered, a means of obtaining the robot's position and orientation as well as sufficiently dense distance measurements is required. The already mentioned *Vectree* system did not have to contend with the high intensity ambient lighting often present in outdoor scenery, and therefore relied on depth (RGB-D) images for ranging information. There are multiple advantages inherent in this approach – depth images are, in most systems, synchronized and overlapping with the color image channel by construction. For some cameras, the color image is directly used in constructing the depth cloud, which means their projective parameters are identical. However, this isn't always the case and the cameras deployed for testing *Vectree* require the intrinsic and extrinsic parameters of each channel to be processed separately. A marked disadvantage for both types of depth cameras, however, is their reliance on structured illumination patterns emitted by the device. These limit the camera's effective range and are easily washed out by the sun outdoors. With these limitations in mind, the data modality of choice for large-scale mapping outdoors is a combination of LiDAR for ranging and RGB for performing semantic inference. EDI has developed a portable sensor package for collecting this type of data and already deployed it in collecting the EDI-SLAM data set [16]. Data in a similar format are also available in other public data sets – for example, *RELLIS-3D* [17], which contains not only the raw images and scans, but also high-frequency ground truth pose estimates and highly detailed point-wise discrete semantic labels.

6.2.1 Overall architecture

The sparsity inherent in mechanically scanned LiDAR point clouds renders surface reconstruction-based map formats such as surfel clouds impractical for many applications due to wide gaps between scan lines. Implicit spatial representations such as NRFs do not have straightforward scaling properties and their method of storing data about the volume is entirely opaque, as model parameters. Voxel occupancy grids, which are well understood and widely employed in LiDAR SLAM systems present the most suitable data structure when designing a semantic mapping system for the intended use case. These efficiently store even very large maps, and simplify map update procedures through the log-odds update rule [13].

The sensor modality and fundamental geometric structure of the map motivate the SLAMVDB system architecture, depicted in Figure 6.1. It needs to be able to process images and scans arriving asynchronously, at

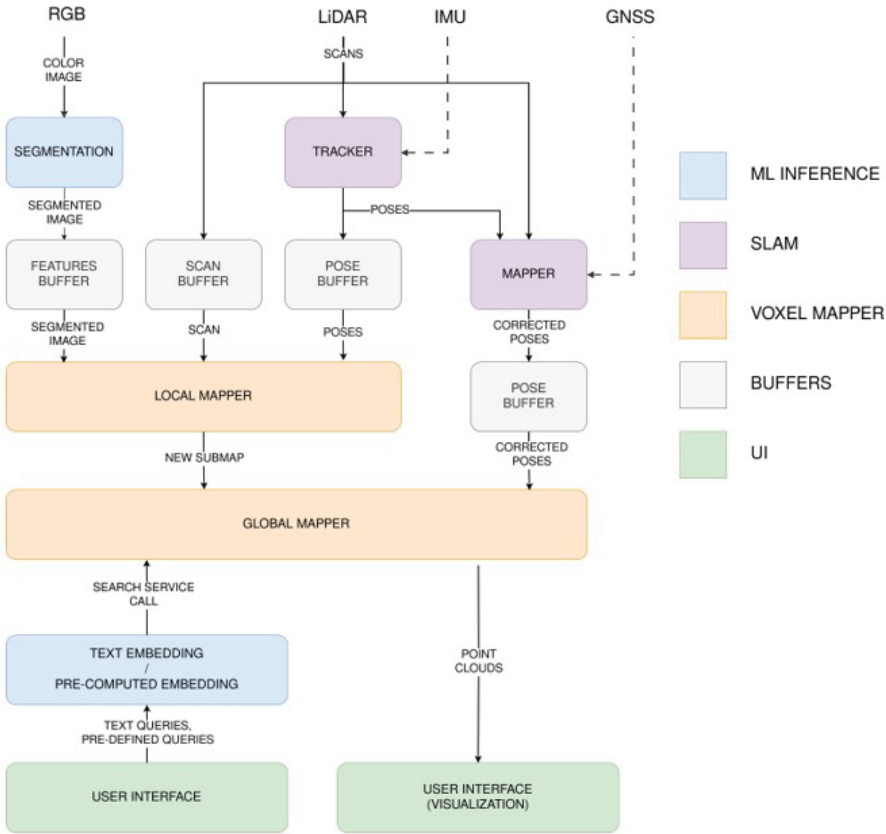

Figure 6.1 Architecture of the SLAMVDB semantic mapping system.

different frequencies, localize the robot, construct a consistent geometric map of the robot's surroundings, and fuse the semantic features extracted from images into this map as they arrive. Hence, the major subsystems in the architecture are the **Tracker** and **Mapper** (loop-closer) for localization; a **Semantic Inference** (image segmentation) module abstracting the neural network-based image processing; and the core of the system – a **Voxel Map**, which stores the environment model and exposes interfaces for lookup operations. Communication between the subsystems – independent software packages in their own right – is implemented using ROS2 [18]. The core voxel mapping application is written in *C++*. The integrated SLAM modules are written in *Python*. ML inference is performed by EDI's fork of *LSeg* [19].

6.2.2 Localization

While purely visual or visual-inertial SLAM systems exist and, under certain circumstances, attain accuracy sufficient to ensure successful robot navigation, LiDAR, which will be available in any deployment or data set SLAMVDB is intended to be used with by construction, still offers greater reliability and tracking accuracy. The SLAMVDB architecture is, in principle, agnostic to the localization provider, but out of the box it ships with two tracker module options and a loop closing mapper. For data sets and deployments where IMU data are available, an integration with the off-the-shelf *fast-lio2* [20] LiDAR-inertial odometry system is provided. No matter what sensor set-up and algorithms are employed for scan-to-scan tracking, error accumulation over time – drift – is expected. To mitigate this, loop closures or position fixes are required. The mapper module exists to accomplish this through pose graph optimization (PGO). This package is decoupled from the tracking module through ROS. It combines the basic ICP-based submap alignment principle employed by *lidarslam_ros2* [21], where a new submap is periodically inserted into a list and compared to previous submaps to detect when the same location has been revisited based on tracker pose estimates, augmented, with a *ScanContext* [22] re-implementation in Python that enables detecting loops even when drift has accumulated too much for raw ICP-based loop detection.

6.2.3 Vector-valued voxel map

The fundamental building block of the semantic voxel map remains the vector-valued octree initially introduced in *Vectree*, though SLAMVDB re-implements it from scratch rather than using an extension of *OctoMap*. The log-odds occupancy update rule remains unchanged. However, the shift from synchronized and co-located RGB-D data for both range and semantics to LiDAR scans combined with RGB from a camera requires a different approach to integrating semantic data. Only a fraction of the points available in every scan will, as a general rule, be directly observed by the camera due to a much more limited field of view and substantial parallax. Hence, during the integration of voxel semantic values $\boldsymbol{v}_{\boldsymbol{x}}$ with the projectively associated image features $\boldsymbol{v}_{\pi(\boldsymbol{x})}$, the existing cell occupancy value reflects the LiDAR observation count, not the visual one, and thus cannot be used directly. Instead, for each visually observed map cell an increment counter $n_{\boldsymbol{x}}$ is implemented, allowing for integration by the rolling average given in Equation 1.1.

$$\boldsymbol{v_x} \leftarrow \frac{n_{\boldsymbol{x}} \cdot \boldsymbol{v_x} + \boldsymbol{v}_{\pi(\boldsymbol{x})}}{n_{\boldsymbol{x}}+1} \tag{6.1}$$

Another fundamental difference is in the projective data association algorithm. For depth cameras with identical depth and color projection parameters, there exists a 1:1 mapping between pixels and points, allowing for a trivial data association. When images and range scans are no longer synchronized, and the voxels grow large relative to image pixels, voxel-to-pixel projection and occlusion modeling is required, which has been implemented in *OpenGL* [23], projecting the integer indices of voxels onto the image plane.

For performance reasons, octree maps are typically constrained in their maximum extent by the need to preserve a bijective mapping between system register width integers and the grid cells. While sufficient for small, confined spaces, when combined with practical voxel resolutions on the scale of centimeters, this approach will fail if the robot traverses many kilometers of wilderness. A more immediate concern, however, is the impact of tracker drift accumulation. If a drift compensation mechanism such as loop closing is employed, it may require changing an already fused voxel map after the fact to reflect an updated trajectory estimate. Hence, rather than building a single unified voxel map, we construct smaller submaps, left free-floating to allow subsequent adjustment. SLAMVDB's voxel mapper implements this as a sequence of submaps inserted periodically based on the salient trajectory length since the last submap insertion. If a loop closure is found, and the submaps are sufficiently close, they are merged into a single submap. Another change from *Vectree* is a simplification - no tree data structure is required to store the voxel semantics, so long as a unique correspondence with the grid cells exists. Instead, a simple key-value store is maintained, with values inserted cleared whenever a cell in the voxel grid is observed visually, or a cell is cleared in the tree, since most grid cells are never visually observed and require no semantics. To facilitate lookup operations in a global reference frame, another data structure is maintained – a global lookup index. This contains read-only pointers to submap key-value store entries at their latest position, and is employed during retrieval operations for map visualization and search. During retrieval, all voxel entries in each index grid cell have their vector values averaged, weighted by observation count.

6.2.4 Image-space inference

As with the previous *Vectree* system, the fundamental approach to image segmentation remains producing two-dimensional feature maps where individual features are vectors in the latent text-image embedding space. Analysis of

the *conceptfusion* algorithm used in *Vectree* reveals that a significant and non-negotiable contribution to the 2-4 second image processing time is by repeated embedding model invocations on the large number of masks proposed by the first model stage. To obtain higher practical frame rates, a system using a single neural network pass was deemed necessary. SLAMVDB uses *LSeg*, and in all practical tests has been configured to operate at a much lower feature map resolution of 128×192 pixels. The resulting system is able to attain a frame rate of 3-5Hz on a consumer-grade laptop computer. Another performance advantage of using *LSeg* is the lower dimension of the embedding vectors – 512 deep rather than *conceptfusion's* 768. However, this difference also naturally means that the text embedding models used by the respective systems are mutually incompatible, as are any query vectors optimized for lookup operations.

6.2.5 Query vector generation

Vectree, was specifically designed to interface with a natural language instruction parsing framework, and therefore process arbitrary text queries. No additional processing was performed – object descriptions were directly embedded and submitted to the vector map for lookup operations. Moreover, only a single query needed to be considered at time. As shown in Figure 6.2, it is possible to hand-craft sets of positive and negative queries whose vector sum, when compared with the image feature map produced by *LSeg*, clearly distinguishes certain terrain types from the background in an image across multiple resolution scales. However, directly using such text-based queries for terrain segmentation across multiple classes leads to the issue observed in Figure 6.3. The absolute similarity scores w.r.t. one query may completely drown out those computed for other lookup vectors, resulting in essentially random results as a single lookup vector hides results from all the others.

To address this issue, SLAMVDB requires a query vector optimization step, depicted in Figure 6.4. Without altering the image or text embedding model weights, a set of query vectors is initialized with class names drawn from the ontology definition. The vectors are then treated as a model parameter matrix and optimized on a relatively small data set using gradient descent in *PyTorch* [24]. This constitutes a classic supervised learning problem in the image segmentation task, but the number of parameters to be optimized is very small – n x 512 where n is the number of classes in the ontology. For the query vectors used in the evaluations in the next section, tracks *00001* and *00002* from the *RELLIS-3D* data set were sampled – 3234 labelled images,

2911 used for training and the rest for validation. The subsequent evaluations were then performed on the other three tracks in the data set, not present for training the query vectors, performing the lookup operations in voxel space. The loss function used is the same as in *LSeg*. The query vectors shipped with the system and used in the evaluation were trained for two epochs.

In terrain segmentation problems, the approach typically taken to produce the class set involves crafting a coarse ontology – grouping the original class labels in a data set into broad traversability classes [15]. For training the query vectors and testing SLAMVDB, the mapping between the *RELLIS-3D* labels ("fine" labels) and the final ("coarse") classes was:

- Smooth – asphalt, concrete;
- Rough – dirt, grass;
- Bumpy – mud, rubble;
- Obstacle – tree, pole, water, object, building, log, fence, bush, barrier;
- Active – vehicle, person;
- Void – void, sky.

6.3 Experimental Assessment and Results

A number of experimental performance assessments were performed across the various stages of the SLAMVDB semantic mapping pipeline, most of which fall outside the scope of this article. Namely, the localization accuracy tests are summarized in a prior technology report document [25]. This section is concerned with the methods used to estimate terrain segmentation accuracy and results achieved in this domain – independently of the localization provider used.

6.3.1 Data set, deployment platform and performance on the edge

While the EDI-SLAM data set [16], collected by EDI using a portable sensor package, was used extensively during the development of SLAMVDB and in testing the accuracy of the SLAM system, as of the writing of this article no semantic annotations are available. For this reason, the *RELLIS-3D* data set, which contains highly detailed point-wise semantic labels, was used in terrain segmentation accuracy assessment. These labels are available on a per-scan basis – in the sensor-local reference frame – meaning that, for methods like SLAMVDB that perform semantic inference on large-scale fused maps rather than on a per-scan basis, a protocol for associating these local point labels

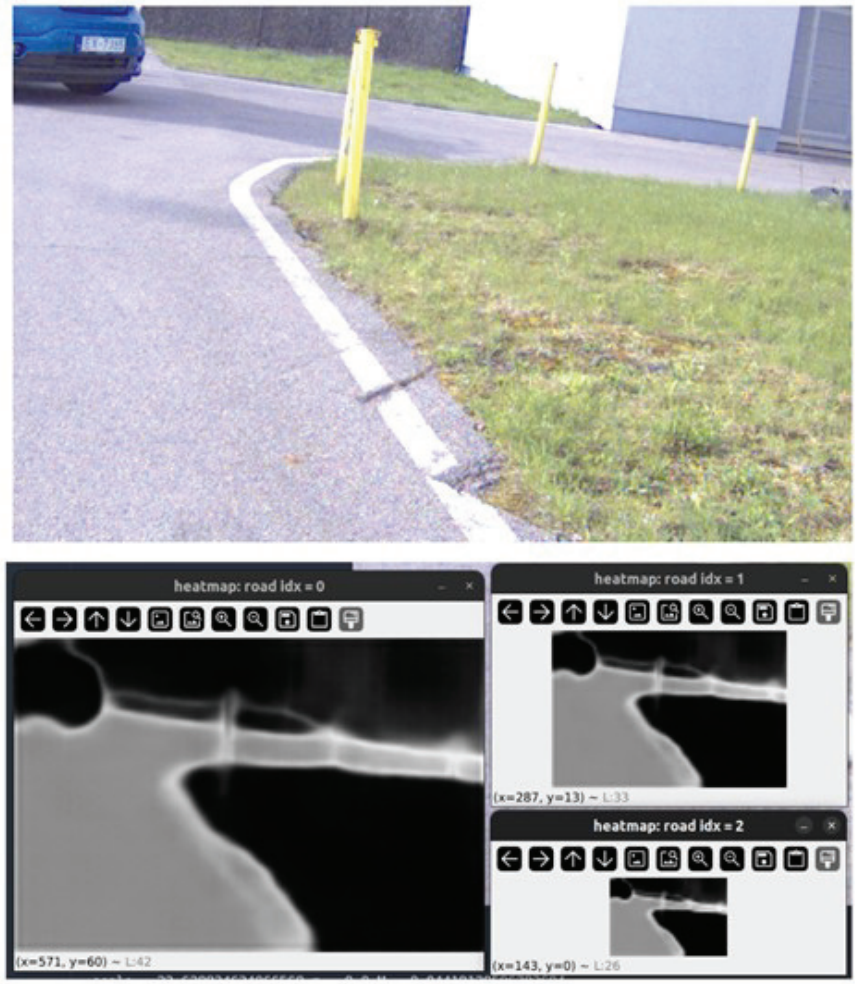

Figure 6.2 Terrain type differentiation using positive and negative text embedding queries.

with global points is required. Since even state of the art LiDAR-inertial tracking systems such as *fast-lio2* accumulate drift on the order of meters over the course of a typical RELLIS-3D track with respect to the provided ground truth trajectory, the pose estimates from such a system cannot be used in measuring the semantic accuracy of the map – the drift will cross voxel boundaries rendering a comparison invalid. Instead, SLAMVDB was configured to use the ground truth trajectory as its localization source. The LiDAR and RGB data were played back in real time using the ROS bag files

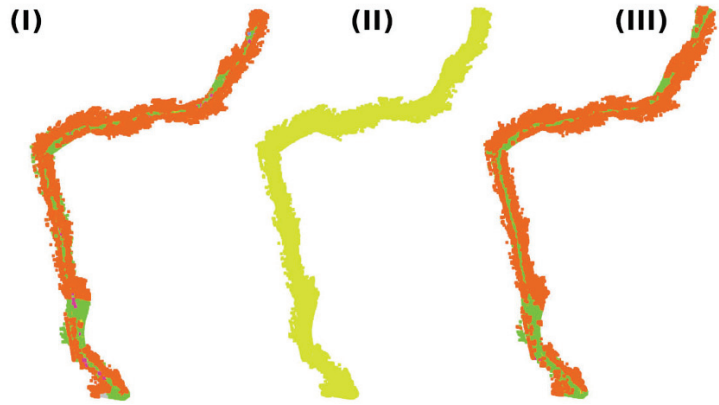

Figure 6.3 (I) The ground truth. (II) Text embeddings. (III) Optimized queries.

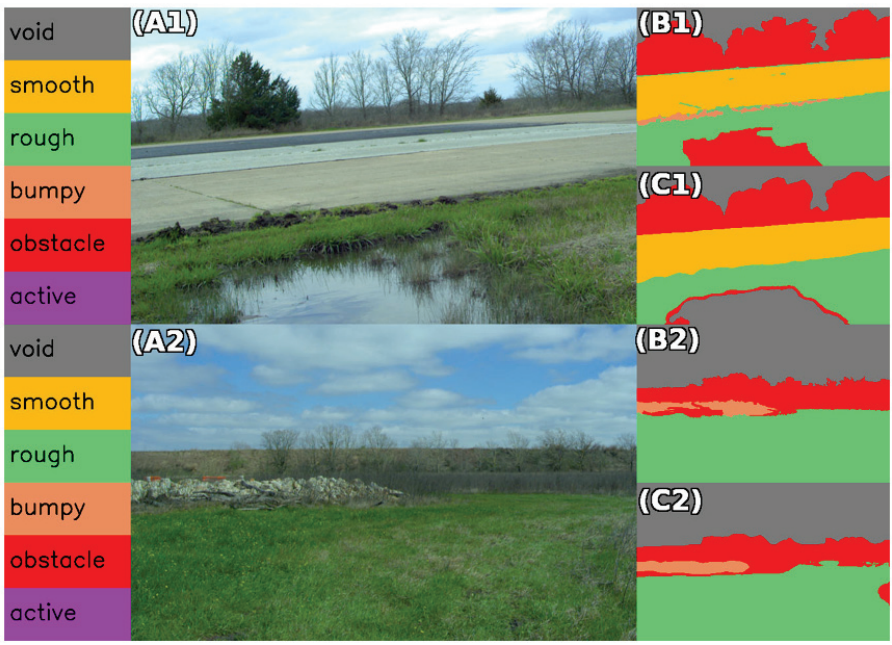

Figure 6.4 (A1-2) RGB images. (B1-2) Ground truth. (C1-2) Inferred classes.

provided as part of the *RELLIS-3D* data set, and the software package was deployed on the same target hardware – a consumer grade laptop: Lenovo Legion 5 with an NVIDIA GeForce RTX 3060m GPU, an AMD Ryzen 5

5600H CPU and 64 GB of RAM. All SLAMVDB software modules were run on this device, which is reflective of the type of deep edge computing hardware commonly found on UGV platforms.

6.3.2 Terrain segmentation accuracy

An issue that must be addressed when comparing a labelled point cloud with a voxel grid map is the choice of discretization. Each point has a theoretically continuously valued set of *x,y,z* coordinates whereas the voxel map forms a discrete grid with cells of finite size. We have performed two different assessments: point-wise accuracy, which measures the probability that any given point in the ground truth data will be inside a grid cell of the same class, and voxel-wise ground truth class assignment, which attempts to assign each voxel in the global reference frame a "correct" label based on the labels that fall within its confines.

Specifically, the point-wise accuracy statistic (heretofore called "hit rate") is expressed as follows:

$$\rho_{hit}, \% = \frac{|\{\text{labelled points in voxel of the same class}\}|}{|\{\text{labelled points in any non}-\text{empty map voxel}\}|} \cdot 100 \tag{6.2}$$

For the voxel-wise labelling, we implement the following procedure:

1. For each non-empty voxel in the map, initialize an instance counter for each class;
2. Transform every scan in the labelled data set into the global reference frame using ground truth poses;
3. For every transformed point, if it falls inside a non-empty voxel, increment the counter for the corresponding class label in that voxel;
4. For each non-empty voxel in the map, assign the class with the highest instance count.

The result of this procedure is a voxel map geometrically identical to the one being assessed, but with classes assigned according to the most frequent point label to be observed within each voxel. For this, we compute three typical set partition statistics – recall, precision and intersection over union (IoU).

The results of this evaluation procedure are collated in Table 6.1. The major row blocks in this table – 00000, 00003, 00004 – correspond to the data set tracks used in the evaluation. For each track, the rows correspond to the classes in the coarse ontology, with the "all" row denoting the overall voxel-wise accuracy (in this case equal to recall, precisions and IoU). The

Table 6.1 Terrain segmentation accuracy (ignoring *Void*)

Track	Class	Hit %	Recall %	Prec. %	IoU %	Recalled (x1000)	Relevant (x1000)
00000	*all*	81.26		87.35		108.42	108.42
	smooth	0.00	0.00	0.00	0.00	0.00	0.16
	rough	75.57	80.26	86.60	71.39	38.83	41.90
	bumpy	4.60	5.34	13.00	3.94	0.30	0.73
	obstacle	89.40	93.64	88.11	83.14	69.26	65.17
	active	15.68	2.67	38.71	2.56	0.03	0.45
00003	*all*	80.39		81.37		60.25	60.25
	smooth	67.70	58.60	90.04	55.04	0.26	0.40
	rough	92.76	92.34	81.14	76.02	39.57	34.77
	bumpy	5.08	6.00	7.43	3.43	0.61	0.75
	obstacle	64.56	71.09	84.02	62.62	19.61	23.18
	active	9.85	14.46	81.50	14.00	0.20	1.13
00004	*all*	73.90		76.17		51.40	51.40
	smooth	0.00	0.00	0.00	0.00	0.00	0.51
	rough	90.95	92.03	74.46	69.95	36.19	29.28
	bumpy	10.32	13.27	49.74	11.70	0.96	3.60
	obstacle	58.98	67.09	82.15	58.55	14.13	17.30
	active	6.09	16.97	96.77	16.88	0.12	0.71

rightmost two columns contain the number of voxels recalled (assigned to a class by inference) and relevant (assigned to a class by the ground truth label) for each of the classes, in each of the tracks, expressed in thousands.

The *Void* class, corresponding to concepts not physically observable using a LiDAR sensor, has been neglected in this assessment. Instead, the second highest similarity class is used for voxels initially classified as *Void*. This heuristic was introduced after observing that transparent objects such as tree canopies tend to pick up the semantics of objects visible through them. In most cases this is non-trivial to address but the *Void* class can be eliminated by construction, improving the highest voxel-wise accuracy score from 84.69% to 87.35% on track 00000. Also apparent is that some classes are represented at very low frequencies, and these also tend to have poor recall metrics even if precision is high. Attempts at addressing this particular issue would most likely benefit from increased image feature resolution, reducing the incidence

of misclassified voxels in image space or projection error. Differences in accuracy between tracks can sometimes be explained by qualitative observations. Track 00000 is the most uniform. It also features the two terrain types that the system appears to be the most successful at distinguishing – grass and trees. Track 00004, on which the lowest accuracy is observed, features a substantial patch of muddy terrain – belonging to the "bumpy" class – that gets classified as a mix of "smooth" and "rough" terrain.

Figure 6.5 depicts this assessment. Items (III) and (IV) in the figure depict the results for a single scan. Item (V) is the set of voxels that has thus far already been tagged with the most frequent ground truth label. Item (VI) is the semantic map produced by SLAMVDB. Also visible in this last section are the LiDAR points that have not been visually observed (in grey). No semantic assignment is possible for these using image segmentation, so SLAMVDB does not store them in the semantic voxel key-value store. They

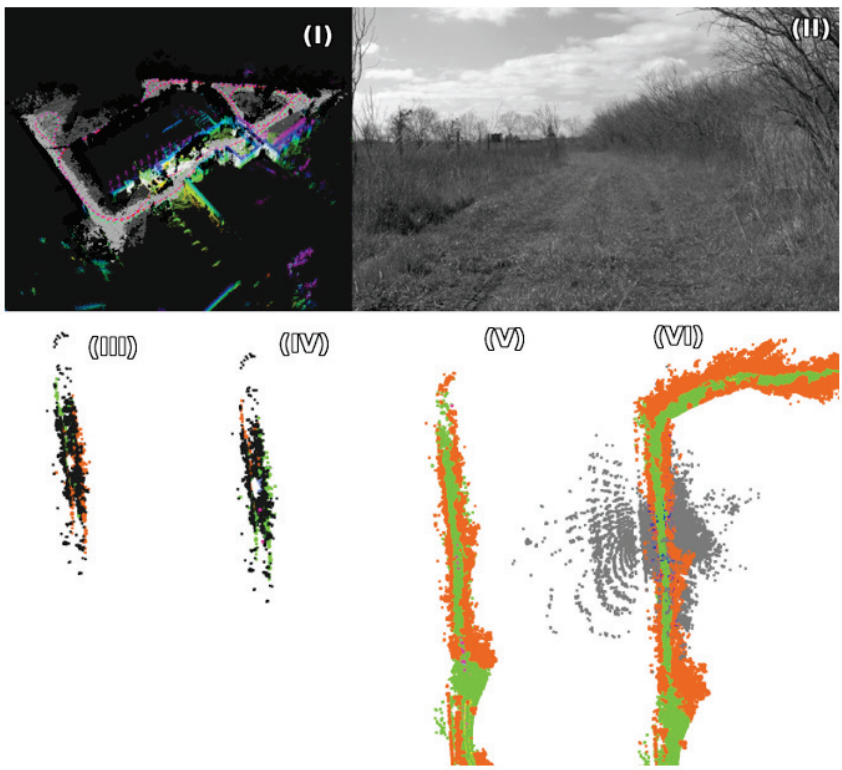

Figure 6.5 Terrain segmentation tests.

do nevertheless contribute to the octree occupancy grid map. Item (I) is a screen capture of SLAMVDB's semantic output on a different data set – the *courtyard_no_gt* track in EDI-SLAM.

6.4 Conclusions

SLAMVDB, the system described in this paper, is a continuation of earlier research in *Vectree*, and successfully demonstrates that the same fundamental approach – storing latent space embeddings in voxel cells and performing search in voxel-space – can be successfully adapted to an entirely different task. Achieving this, however, has (at least so far) taken some adaptations that depart somewhat from the idealized notion of a zero-shot generalizable classifier. Specifically, to address issues with different scaling for similarity distributions of the map semantics w.r.t. different query vectors, a query vector training procedure had to be introduced. This is still substantially less time- and resource-intensive than training a neural network model – requiring only that a handful of vectors be adjusted, and usable with a small data set of labelled images.

Qualitatively, the system appears quite capable of distinguishing between most obstacles and flat, passable terrain, though substantial room for improvement remains. An obvious direction for future work is the integration of newer, more capable image embedding model architectures, such as YOLOE [26]. Other directions already being actively pursued are deployment on real-world UGV platforms for testing in the field, integration of data from multiple UAV, UGV agents, and integration into 3-dimensional scanning software solutions, with one study underway in the space of monitoring and surveying of construction sites.

Acknowledgments

This research was partially supported by: "EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing", CHIPS JU grant agreement No 10109730".

References

[1] M. Ryan, 'Labour and skills shortages in the agro-food sector', Organisation for Economic Co-operation and Development (OECD), Paris, 2023.

[2] J. Arents and M. Greitans, 'Smart Industrial Robot Control Trends, Challenges and Opportunities within Manufacturing', 2022.
[3] R. Keith and H. M. La, 'Review of Autonomous Mobile Robots for the Warehouse Environment', ArXiv, vol. abs/2406.08333, 2024.
[4] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva, 'Advances in agriculture robotics: A state-of-the-art review and challenges ahead', Robotics, vol. 10, no. 2, p. 52, 2021.
[5] P. Racinskis, J. Ārents, and M. Greitans, 'Constructing Maps for Autonomous Robotics: An Introductory Conceptual Overview', Electronics, 2023.
[6] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, 'SemanticFusion: Dense 3D semantic mapping with convolutional neural networks', in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 4628–4635.
[7] A. Radford et al., 'Learning Transferable Visual Models From Natural Language Supervision', in International Conference on Machine Learning, 2021.
[8] B. Li, K. Q. Weinberger, S. J. Belongie, V. Koltun, and R. Ranftl, 'Language-driven Semantic Segmentation', ArXiv, vol. abs/2201.03546, 2022.
[9] K. M. Jatavallabhula et al., 'ConceptFusion: Open-set Multimodal 3D Mapping', ArXiv, vol. abs/2302.07241, 2023.
[10] N. Ravi et al., 'SAM 2: Segment Anything in Images and Videos', ArXiv, vol. abs/2408.00714, 2024.
[11] K. Mazur, E. Sucar, and A. J. Davison, 'Feature-Realistic Neural Fusion for Real-Time, Open Set Scene Understanding', ArXiv, vol. abs/2210.03043, 2022.
[12] P. Racinskis, O. Vismanis, T. E. Zinars, J. Arents, and M. Greitans, 'Towards Open-Set NLP-Based Multi-Level Planning for Robotic Tasks', Applied Sciences, vol. 14, no. 22, p. 10717, 2024.
[13] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, 'OctoMap: an efficient probabilistic 3D mapping framework based on octrees', Autonomous Robots, vol. 34, pp. 189-206, 2013.
[14] P. Racinskis, 'SLAMVDB - the EDI SLAM Vector Data Base (prerelease version)'. [Online]. Available: https://github.com/edi-administrator/SLAMVDB [Accessed: 15-Oct-2025].
[15] T. Guan, D. Kothandaraman, R. Chandra, and D. Manocha, 'GANav: Group-wise Attention Network for Classifying Navigable Regions

in Unstructured Outdoor Environments', ArXiv, vol. abs/2103.04233, 2021.

[16] P. Racinskis, G. Krasnikovs, J. Arents, and M. Greitans, 'The EDI Multi-Modal Simultaneous Localization and Mapping Dataset (EDI-SLAM)', Data (2306-5729), vol. 10, no. 1, 2025.

[17] P. Jiang, P. R. Osteen, M. B. Wigness, and S. Saripalli, 'RELLIS-3D Dataset: Data, Benchmarks and Analysis', 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 1110-1116, 2020.

[18] S. Macenski, T. Foote, B. P. Gerkey, C. Lalancette, and W. Woodall, 'Robot Operating System 2: Design, architecture, and uses in the wild', Science Robotics, vol. 7, 2022.

[19] P. Racinskis, 'LSeg: trimmed down fork for use with EDI-SLAMVDB'. [Online]. Available: https://github.com/edi-administrator/langseg-public [Accessed: 15-Oct-2025].

[20] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, 'FAST-LIO2: Fast Direct LiDAR-Inertial Odometry', IEEE Transactions on Robotics, vol. 38, no. 4, pp. 2053-2073, 2022.

[21] rsasaki0109, 'lidarslam_ros2'. [Online]. Available: https://github.com/rsasaki0109/lidarslam_ros2 [Accessed: 15-Oct-2025].

[22] G. Kim and A. Kim, 'Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map', 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4802-4809, 2018.

[23] Khronos Group, The OpenGL Specification, Version 4.6, July 2017. Available: https://www.khronos.org/registry/OpenGL/index_gl.php [Accessed: 05-Dec-2025]

[24] A. Paszke et al., "Automatic Differentiation in PyTorch," in Proc. NIPS Workshops, 2017. Available: https://openreview.net/pdf?id=BJJsrmfCZ [Accessed: 04-Dec-2025].

[25] M. Greitans et al. "RoLISe T4.1 Published materials". [Online]. Available: https://www.edi.lv/RoLISe_T4_1 [Accessed: 15-Oct-2025].

[26] A. Wang, L. Liu, H. Chen, Z. Lin, J. Han, and G. Ding, 'YOLOE: Real-Time Seeing Anything', arXiv [cs.CV]. 2025.