Collaborative Dynamic 3D Scene Graphs for Automated Driving

Elias Greve[1*], Martin Büchner[1*], Niclas Vödisch[1*], Wolfram Burgard[2], and Abhinav Valada[1]

Abstract— Maps have played an indispensable role in enabling safe and automated driving. Although there have been many advances on different fronts ranging from SLAM to semantics, building an actionable hierarchical semantic representation of urban dynamic scenes from multiple agents is still a challenging problem. In this work, we present Collaborative URBan Scene Graphs (CURB-SG) that enable higher-order reasoning and efficient querying for many functions of automated driving. CURB-SG leverages panoptic LiDAR data from multiple agents to build large-scale maps using an effective graph-based collaborative SLAM approach that detects inter-agent loop closures. To semantically decompose the obtained 3D map, we build a lane graph from the paths of ego agents and their panoptic observations of other vehicles. Based on the connectivity of the lane graph, we segregate the environment into intersecting and non-intersecting road areas. Subsequently, we construct a multi-layered scene graph that includes lane information, the position of static landmarks and their assignment to certain map sections, other vehicles observed by the ego agents, and the pose graph from SLAM including 3D panoptic point clouds. We extensively evaluate CURB-SG in urban scenarios using a photorealistic simulator. We release our code at `http://curb.cs.uni-freiburg.de`.

I. INTRODUCTION

Spatial and semantic understanding of the environment is crucial for the safe and autonomous navigation of mobile robots and self-driving cars. Recent autonomy systems leverage high-definition (HD) map information as effective priors for several downstream tasks in automated driving (AD) including perception [1], localization [2], planning [3], and control [4]. HD maps are often constructed and maintained in a top-down manner [5], i.e., relying on traffic authorities or via arduous labeling efforts. In contrast, automatic bottom-up AD mapping approaches show high accuracy [6], [7] while being limited to occupancy or semantic mapping using, e.g., dense voxel grid manifolds. With respect to AD, map representations should ideally fulfill the following requirements [8]: 1) completeness and accuracy while scaling to large areas; 2) frequent updates to capture structural changes; 3) higher-level topological information grounded in rich sensor data; 4) efficient access and information querying. Given these requirements, typical SLAM maps only enable classical spatial or point-level semantic querying. We envision that modern AD mapping approaches should provide the means to process vision and language queries, e.g., from foundation models [9]. Enabling such demands can only
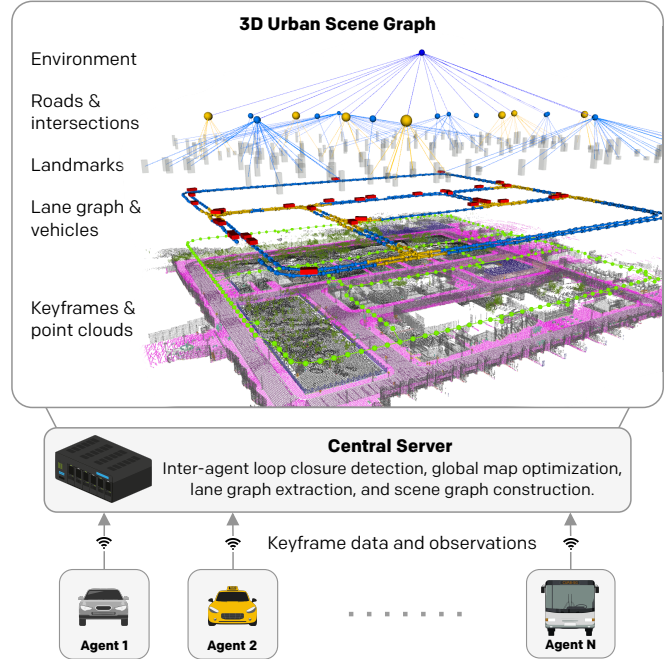

Fig. 1. For our proposed collaborative urban scene graphs (CURB-SG), multiple agents send keyframe packages with their local odometry estimates and panoptic LiDAR scans to a central server that performs global graph optimization. We subsequently partition the environment based on a lane graph from agent paths and other detected cars. Together with the 3D map, the lane graph forms the base of the large-scale hierarchical scene graph.

become feasible by abstracting from given maps using sparse representations.

In this work, we propose *Collaborative URBan Scene Graphs* (CURB-SG) that effectively address the aforementioned requirements by constructing a hierarchical graph structure of the environment as shown in Fig. 1. 3D scene graphs enable efficient data storage of large environments while being queryable and preserving spatial information. Previous works on 3D scene graphs [10], [11], [12] focus on indoor environments, whose taxonomy cannot be directly transferred to large-scale urban domains. To close this gap, we introduce the following analogy to indoor variants: Cities (buildings) can be separated into intersections and roads (rooms), which contain static landmarks such as traffic signs (furniture) as well as dynamic objects such as vehicles (humans). We enable this partitioning by generating an online lane graph that serves as a common link among multiple graph layers. Addressing frequent updates and multi-agent cooperation, our method leverages a centralized collaborative SLAM approach that combines panoptic LiDAR data and local odometry estimates into a single 3D map while optimizing a global pose graph that benefits from inter-agent

---

* Equal contribution.
[1] Department of Computer Science, University of Freiburg, Germany.
[2] Department of Eng., University of Technology Nuremberg, Germany.

loop closures. Following the spirit of previous works on scene graphs [10], [11], [12], we extensively evaluate our proposed method on simulated data using the CARLA simulator [13].

To summarize, the main contributions are as follows:

1) We introduce a novel algorithm for representing urban driving environments as dynamic 3D scene graphs that are constructed from multi-agent observations to efficiently cover large areas.

2) We demonstrate an effective partitioning of urban environments using lane graphs constructed on the fly from panoptic LiDAR observations in a cooperative manner.

3) We present an efficient collaborative graph SLAM method to continuously update semantic maps while addressing scalability via edge contraction.

4) We provide extensive evaluations of the building blocks of our proposed framework.

5) We make our code and sample data publicly available at http://curb.cs.uni-freiburg.de.

II. RELATED WORK

In this section, we first present a summary of LiDAR-based odometry and mapping, followed by an overview of multi-agent SLAM, and scene graphs in automated driving (AD).

LiDAR SLAM: LiDAR-based mapping has been pioneered by LOAM [14] that estimates robot motion from scan registration via ICP between subsequent point clouds. To address the full SLAM problem, HDL Graph SLAM [7] combines LiDAR odometry with local loop closure detection and performs joint pose graph optimization. Leveraging semantic segmentation, SUMA++ [6] masks dynamic classes during the mapping stage and proposes a semantic-aided variant of ICP. PADLoC [15] exploits panoptic segmentation during training to stabilize both loop closure detection and registration. In this work, we use panoptic point clouds to generate a large-scale semantic 3D map forming the base layer of our scene graph.

Collaborative SLAM: To cover large environments and to increase mapping speed, SLAM research begins to shift towards multi-agent methods [16]. Generally, collaborative SLAM can be realized in a centralized or distributed manner. Initial works such as C^2TAM [17] belong to the centralized category, performing global bundle adjustment on a server and localization on the clients. A similar paradigm is adopted by CVI-SLAM [18] and COVINS [19], proposing visual-inertial (VI) SLAM systems for a fleet of UAVs. While the robots run local VI odometry, a central server collects this information, searches for inter-agent loop closures to perform global optimization, and removes redundant data. With respect to LiDAR SLAM, LAMP 2.0 [20] allows collaboration between different types of robots to map large-scale underground environments. A similar use case is addressed by Swarm-SLAM [21], which supports further sensor modalities. Following a distributed paradigm, information is directly shared between the agents using peer-to-peer communication. Kimera-Multi [22] is a VI SLAM method that includes semantic information in the generated 3D mesh. For data fusion, it employs distributed pose graph optimization (PGO). Finally, DisCo-SLAM [23] proposes a LiDAR-based approach addressing the initially unknown relative position of the agents. For this, they use Scan Context [24] descriptors for global loop closure detection without spatial priors. In this work, we follow the centralized paradigm since we leverage collaborative SLAM to generate a single consistent scene graph that can be made available to other traffic participants to query information.

Scene Graphs for Automated Driving: 3D scene graphs constitute an effective interface unifying pose graphs from large-scale mapping and local information [25] such as frame-wise object detections [26], topological mapping [27], [28], [29], or semantic segmentation [30], [31]. Additionally, graphs enable the structural disassembly of large-scale scenes into objects and their relationships and facilitate higher-level reasoning, e.g., in the vision and language domain [32]. This further allows for efficient hierarchical abstraction in both spatial and semantic regimes [12], [33]. So far, 3D scene graphs for environment representation have only been applied in indoor domains. The first work in this field [12] proposes an offline, multi-layered hierarchical representation based on RGB images. Kim *et al.* [34] were the first to generate 3D scene graphs from RGB-D images for visual question answering [35] and task planning. Using a learning-based pipeline, Wald *et al.* [36] construct a 3D scene graph from an instance-segmented point cloud while predicting node and edge semantics in an offline manner. Rosinol *et al.* [33] present an offline framework capable of generating hierarchical scene graphs from dynamic indoor scenes that are divided into buildings, rooms, places, objects, and agents, as well as a metric-semantic mesh. Different from the aforementioned frameworks, Hydra [11], SceneGraphFusion [37], and S-Graphs [25] present real time-capable approaches. While Hydra does not tightly couple the optimized pose graph with the 3D scene graph, the non-hierarchical S-Graphs [25] close this gap. The follow-up work S-Graphs+ [10] also encodes hierarchies. In this work, we combine collaborative SLAM and 3D scene graphs to build hierarchical maps for AD. To the best of our knowledge, our work constitutes the first approach to 3D scene graph construction of urban driving scenes with a tightly coupled integration of inter-agent loop closures. Furthermore, we show how multi-agent cooperation facilitates frequent map updates and completeness.

III. TECHNICAL APPROACH

In this section, we present our CURB-SG approach for collaborative urban scene graphs. As illustrated in Fig. 2, CURB-SG is comprised of several components. In Sec. III-A, we describe our approach for collaborative SLAM to effectively combine panoptic information. Here, multiple agents transmit their onboard LiDAR odometry estimates along with panoptic point clouds to a central compute unit. This server combines the data by detecting intra- and inter-agent loop closures and performs pose graph optimization (PGO) to generate a globally consistent 3D map. In Sec. III-B, we propose to further aggregate the paths of the agents
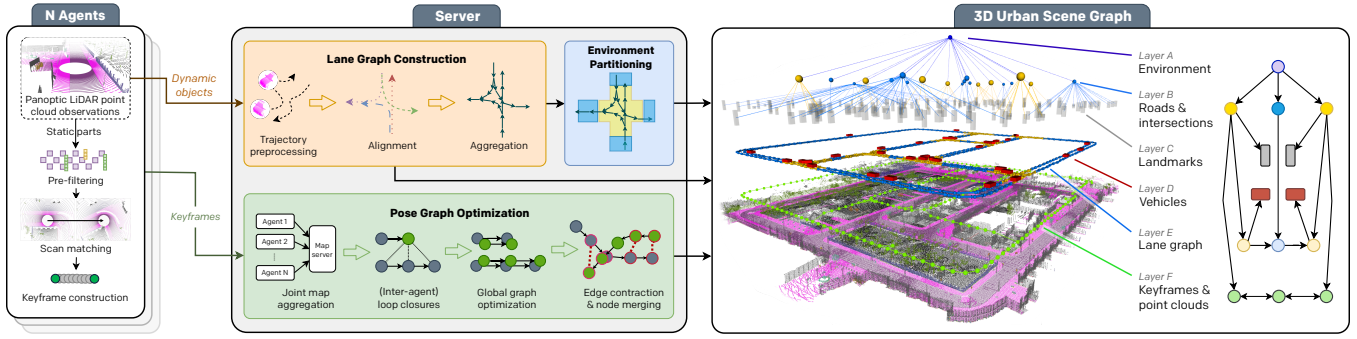
Fig. 2. Overview of CURB-SG: Multiple agents obtain panoptically segmented LiDAR data and provide an odometry estimate based on the static parts of the point cloud. A centralized server instance then performs pose graph optimization (PGO) including inter-agent loop closure detection and edge contraction based on the agents' inputs. Tightly coupled to the pose graph, we aggregate a lane graph from panoptic observations of other vehicles as well as the agent's trajectories. Next, the lane graph is partitioned to retrieve a topological separation that allows for the hierarchical abstraction of larger environments.

and other observed vehicles to extract an online lane graph allowing for partitioning the city into intersections and roads. Finally, the server registers dynamic traffic participants on the lane graph and generates a hierarchical scene graph by assigning static landmarks to the closest intersection or road.

A. Collaborative SLAM

We leverage collaborative LiDAR SLAM as the backend in our proposed CURB-SG. Due to its reliable performance and well-maintained code base, we build on top of HDL Graph SLAM [7] and extend it to a multi-agent scenario following a centralized approach as described in Sec. II. In this section, we describe the steps performed by each agent, followed by the centralized PGO as depicted in Fig. 2. Finally, we provide further details on how CURB-SG explicitly addresses both long-term and large-scale mapping.

Agents: Each agent is equipped with a LiDAR sensor to capture sparse 3D point clouds, which contain spatial information as well as point-wise panoptic segmentation labels. Initially, a point cloud is separated into its static and dynamic components following the conventional categorization of "stuff" and "thing" classes [38]. Similar to SUMA++ [6], we use only the static points for constructing the map. In contrast to HDL Graph SLAM [7], we utilize different voxel grid sizes for the various semantic classes. This approach retains more dense information where required, e.g., poles and traffic signs are being processed at a more fine-grained level than roads or buildings. Next, we perform point cloud registration via FAST-GICP [39] between subsequent LiDAR scans to estimate the motion of an agent. Following the common methodology and to reduce the required bandwidth between the agents and the server, we generate keyframes after a specified traveled distance based on LiDAR odometry. Each keyframe is sent to the server and contains an estimated pose and the static LiDAR point cloud with semantic labels, i.e., the "stuff" points. Since car instances contribute to the online construction of a lane graph (see Sec. III-B), the "thing" points from all the LiDAR scans are transformed relative to the pose of the previous keyframe and sent separately.

Server: The centralized server receives keyframes from all the agents and processes them in the following manner: First,

upon receiving the first keyframe sent by an agent, the server registers this agent to the global pose graph. Second, the server searches for loop closure candidates between the added keyframe and the existing nodes in the pose graph to find both intra- and inter-agent loop closures. We rely on the original loop closure detection technique of HDL Graph SLAM [7], i.e., all nodes within a local search radius are considered to be candidates. If the fitness score of the ICP algorithm is below a threshold, a loop closure edge is added to the pose graph. Due to relying on an initial guess, we utilize the absolute ground truth value for the registration of a new agent. In practice, this could either be solved with GNSS measurements or by conducting an efficient global search for loop closure candidates leveraging point cloud descriptors [23]. Third, the server performs PGO using g^2o [40] to integrate the newly added keyframes and detected loop closures. To address scalability, we employ edge contraction as detailed in the following paragraph. Finally, we apply the same semantics-based voxelization to the entire 3D map as performed by the agents on their local LiDAR scans.

Long-Term and Large-Scale Mapping: If not handled explicitly, the pose graph would continue to grow while the mapping progresses. Since every keyframe contains a 3D point cloud, this not only significantly slows down the PGO but also increases memory consumption and disk storage. To address both problems, we remove the nodes and edges from the graph that carry redundant information. In Fig. 3, two agents have driven along the same road yielding multiple loop closures. Using a heuristic-driven approach, the loop closure edges that carry redundant information are being contracted by merging nodes. By redirecting the edges of the omitted to the remaining node, we ensure the legal connectivity of the pose graph. Notably, this is done after the PGO step. Consequently, the final pose graph becomes easier to maintain and more efficient to query when searching for new loop closures. The point cloud data associated with a removed node is combined with the data of the persisting node while omitting older data to guarantee up-to-date map information. In contrast, the dynamic observations linked to a node are completely transferred as they contribute towards the construction of the lane graph explained in Sec. III-B. For the same reason, each
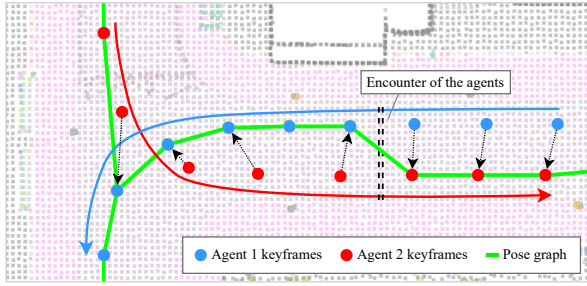
Fig. 3. In this example, two agents drive along the same road while passing each other at the dashed line. The detected loop closures yield additional edges in the pose graph. After optimization, the edges that carry redundant information are contracted by merging the older node into the more recently added node to update the map information.

removed node is turned into a passive observation that stores the driven path of an ego agent.

B. Scene Graph Generation

The second key component of CURB-SG is a scalable environment representation of urban outdoor scenes for AD. Besides the aforementioned 3D semantic map, CURB-SG constructs a tightly-coupled hierarchical abstraction of the environment as shown in Fig. 2. By analogy with the separation of indoor scenes into buildings, rooms, and places [33], [12], [11], we decompose a constructed lane graph into intersecting and non-intersecting road areas allowing for spatial and semantic abstraction.

The root of our CURB-SG representation is given by *Layer A* that holds environment/city-level information. This environment is then spatially divided into intersections and their connecting roads (*Layer B*), which serve as the categorical counterparts to rooms and corridors in indoor scenes. Since the partitioning of our environment is based on a lane graph (presented in *Layer E*), the connectivity of *Layer B* is implicitly given by the connectivity of the lane graph (colored segments, Fig. 2). Next, we map static landmarks such as traffic signs and poles contained in *Layer C* including their bounding box to their corresponding spatial area defined by *Layer B*. These landmarks can serve as priors for localization or object detection. *Layer D* holds all currently observed dynamic vehicles. We map dynamic vehicles to their closest respective lane graph node, as defined in *Layer E*, to provide efficient access for downstream tasks, e.g., trajectory prediction. Central to this approach, *Layer E* is a directed lane graph to encode the low-level topology for vehicle navigation and is inferred from the paths of the ego agents as well as other perceived vehicles. We provide further details in the next paragraph. The lane graph defines the connectivity of the different spatial regions in the urban environment, comparable to edges among rooms in indoor scene graph variants. Finally, *Layer F* contains the pose graph from our SLAM backend and encodes LiDAR data in the form of semantic point clouds. As discussed in Sec. III-A, this layer is subject to continuous optimization and dynamic restructuring, e.g., due to loop closure detection and edge contraction. Based on the edges between the keyframes in this layer and spatial areas (*Layer B*), 3D map information is easily accessible given a

rough road-level position estimate.

Lane Graph Generation: We generate a lane graph of the environment leveraging the trajectories of the ego agents as well as observations of surrounding vehicles. As the LiDAR point clouds of the agents contain instance IDs, we are able to differentiate between multiple observed vehicle instances in the agents' surroundings. For each observed vehicle, we extract the centroid of its partial point cloud. The position of a centroid is stored relative to the most recent keyframe. After transmitting the data to the server, the position of this dynamic observation can be retrieved given the link to its corresponding keyframe. Consequently, the positions of all the dynamic observations benefit from continuous keyframe updates due to PGO as depicted in Fig. 2. To evenly sample paths, we further filter the observations using both hand-crafted heuristics and DBSCAN [41] based on timestamps, angles, and relative displacements. This is particularly important for stationary and occluded objects as well as outliers caused by odometry noise. Following an iterative yaw-respective aggregation scheme [27], we convert all trajectories into directed graphs, apply Laplacian smoothing, and merge them to build a complete lane graph. Employing the same processing scheme, we add agent trajectories to this graph. Since CURB-SG maintains a connection between the lane graph and the keyframes used in SLAM, we can continuously propagate refinements from PGO to the lane graph.

Spatial Partitioning: Urban outdoor driving scenes exhibit a vastly different topology compared to indoor environments that have been represented using scene graphs so far. We found that classical methods such as wall dilation for retrieving disjoint environment graphs [11] are not directly applicable to urban environments. In our work, we propose to separate outdoor environments into intersecting and non-intersecting areas using the obtained lane graph (see above). Ultimately, this gives rise to the hierarchical environment abstraction introduced in CURB-SG enabling efficient querying for downstream tasks such as trajectory prediction. In particular, we detect intersections based on the following heuristics: First, we cluster high-degree lane graph nodes to find agglomerations of graph splits and merges. Second, we detect lane graph edges that intersect. These two approaches can be applied to various environments to efficiently handle challenging conditions such as multi-lane roads or non-trivial intersections. After identifying intersection nodes, the remaining disconnected sub-graphs fall into non-intersecting road areas. To assign components from other layers of the scene graph to the extracted partitions, we extend these areas beyond the lane node surroundings as illustrated in Fig. 2.

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate CURB-SG with respect to the collaborative SLAM backend, the constructed lane graph, and the proposed partitioning based on road intersections.

A. Experimental Setup

We evaluate CURB-SG on various urban driving scenarios using the CARLA simulator [13] due to a lack of real-world

Environment	Agent count	RMSE (agents) [m]	RMSE (street signs) [m]	Exploration time [sim. steps]
town01	1	0.735 ± 0.492	0.865 ± 0.485	2502.50
	2	0.368 ± 0.343	0.480 ± 0.358	1267.70
	3	**0.132 ± 0.096**	**0.169 ± 0.096**	**1134.40**
+ noise	3	0.159 ± 0.093	0.225 ± 0.101	–
town02	1	0.306 ± 0.208	0.297 ± 0.194	2079.00
	2	0.249 ± 0.176	0.299 ± 0.238	943.80
	3	**0.126 ± 0.109**	**0.164 ± 0.148**	**597.60**
+ noise	3	0.119 ± 0.078	0.140 ± 0.064	–
town07	1	0.564 ± 0.406	–	3632.70
	2	0.234 ± **0.182**	–	1760.20
	3	**0.218** ± 0.198	–	**910.00**
town10	1	0.333 ± 0.185	–	923.70
	2	0.310 ± 0.185	–	724.00
	3	**0.116 ± 0.106**	–	**391.10**

Mean and standard deviation over ten runs of the RMSE of the agents' keyframes and the estimated position of the street signs representing localization and mapping accuracy, respectively. Note that the environments *town07* and *town10* do not contain street signs. The rightmost column lists the mean time required to map 85 % of the entire town measured in simulation steps.
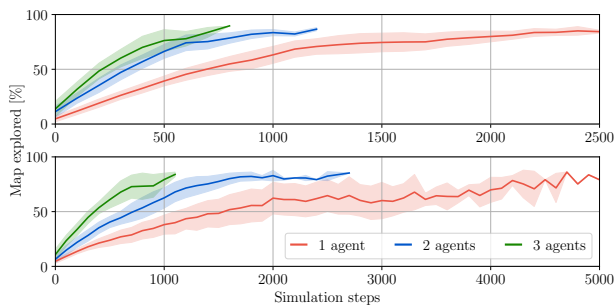

Fig. 4. The mapping progress in *town02* (top) and *town07* (bottom) for one, two, and three agents. Our collaborative SLAM method benefits from receiving inputs from multiple agents.
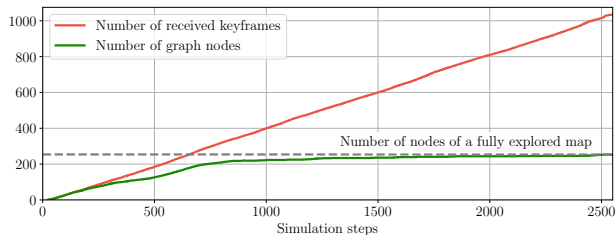

Fig. 5. Our proposed edge contraction mechanism effectively reduces the number of nodes in the pose graph to maintain the capability of frequent graph optimization. This plot shows three agents operating in *town02*.

multi-agent datasets providing LiDAR scans. In particular, we perform experiments on a set of four diverse environments including *town01*, *town02*, *town07*, and *town10*. Following previous works [11], we use the panoptic annotations with temporally consistent instance IDs provided by the simulator. Where applicable, we demonstrate the efficacy of CURB-SG for one, two, and three agents and average results over ten randomly initialized runs. Due to the semantics-based voxelization on the server, the total number of map points of a fully explored town is relatively stable. As the path planning of the agents is randomized, it can take a long time until this number is reached. Therefore, we approximate full exploration by using 85 % as the termination criterion.

B. Collaborative SLAM

In this section, we evaluate the collaborative SLAM backend of our proposed CURB-SG with respect to both accuracy and cooperative gain in long-term scenarios.

Mapping and Localization: In Tab. I, we present the root mean squared errors (RMSE) of the agents' keyframes and the estimated position of the street signs to represent localization and mapping accuracy, respectively. We compute the position of a street sign as the geometric center of the corresponding bounding box that is inferred from the 3D map. We observe that both errors are reduced when more agents contribute towards the collaborative pose graph. Except for the case of two agents in *town07*, this holds true for the mean as well as the standard deviation across all environments. We further illustrate the robustness of our approach against noisy sensor data by imposing realistic metric Gaussian noise $\mathcal{N}(0, 0.02)$ on the LiDAR scans [42] of the agents in *town01* and *town02*. As shown in Tab. I, the noise does not significantly alter the errors indicating that downstream tasks such as lane graph estimation do not degrade either.

Long-Term Mapping: We demonstrate the efficacy of our

proposed adaptions of HDL Graph SLAM [7] (see Sec. III-A) to address long-term mapping of large areas. In the rightmost column of Tab. I, we report the time required to map a town when using one, two, or three agents. Generally, the higher the number of contributing agents, the smaller the time required to explore the map. Similarly, in Fig. 4, we illustrate the mapping progress measured by the number of 3D points versus the simulation steps. While the results confirm the aforementioned general trend towards faster exploration in a multi-agent setup, the pure mapping speed will reach an upper bound above that additional agents will not further increase the speed. However, even afterward, these agents will keep sending measurements and vehicle observations contributing towards frequent map updates and enhancing the lane graph (see Sec. IV-C). We present further results for *town01* and *town10* in the suppl. material Sec. S.3.

Finally, we demonstrate that our proposed edge contraction successfully limits the number of nodes contained in the pose graph. In Fig. 5, we show the example of three agents operating in *town02* and compare the number of optimizable graph nodes with the total number of keyframes sent by the agents. We observe that without edge contraction, the pose graph continuously grows with the number of keyframes sent, rendering frequent optimization infeasible.

C. Lane Graph

We evaluate our proposed online lane graph generation approach from the paths of the ego agents and their observations of other vehicles (Sec. III-B). We present qualitative results in Fig. 6 for two scenarios simulated in *town02* with 30 additional non-agent vehicles: the left figure visualizes the lane graph in a single-agent scenario terminated as soon as

Ego	Obs.	TOPO P / R	GEO P / R	APLS	SDA$_{4.5}$	SDA$_{9.0}$	Graph IoU
1-agent scenario							
✓		**0.810** / 0.281	**0.923** / 0.415	0.658	0.000	0.042	0.386
✓	✓	0.678 / **0.562**	0.855 / **0.812**	**0.724**	**0.278**	**0.394**	**0.690**
3-agents scenario							
✓		**0.715** / 0.583	**0.874** / 0.762	**0.800**	0.188	0.357	0.658
✓	✓	0.574 / **0.712**	0.751 / **0.925**	0.756	**0.250**	**0.452**	**0.751**

Quantitative results obtained in *town02*. The two left columns indicate whether only the paths of the ego agents or also the estimated positions of other observed vehicles have been used. For the TOPO and GEO metrics, we provide both precision (P) and recall (R).

Fig. 7. Intersection detection quality of our lane graph-based detection of intersections (LG) and an image-based skeletonization baseline of the road surface (SK). Average precision (P) and recall (R) of both approaches across 10 runs with 3 agents and 40 vehicles on *town02* as well as the size of the investigated road surface point cloud are shown.
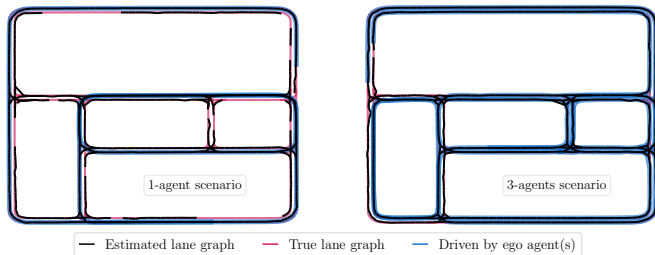

Fig. 6. Visualization of the constructed lane graph of *town02* when using one or three agents. Lanes marked in blue have been traversed by an ego agent. Others are reconstructed from observing surrounding vehicles.

the agent starts to repeatedly revisit intersections. Although the path of the agent, shown in blue, does not cover all the lanes, including the paths of the observed vehicles allows for a substantial extension of the lane graph. The right figure depicts a long-term scenario with three agents demonstrating that collaboration further boosts performance. Our method yields an almost complete lane graph even though several lanes have only been driven by the agents in the opposite direction.

We quantify these findings in Tab. II following previous works on lane graphs: precision and recall of the TOPO and GEO metrics [29], APLS [43], SDA$_R$ [27] with the subscript denoting the search radius in meters, and the graph IoU [27]. For more details, please refer to the respective reference. We observe that except for the TOPO/GEO precision and the APLS in the 3-agent scenario, all the metrics show an improvement when using not only the paths of the ego agents but also of the observed vehicles. We attribute the decrease in precision to the noise in the estimated position of the other vehicles. Since we approximate the center of a vehicle by the geometric mean of the respective 3D points, there is a bias towards the center line of a road for all oncoming cars. We further observe that increasing the number of agents does have a positive impact on all the metrics except for the TOPO/GEO precision and the SDA$_{4.5}$ demonstrating the efficacy of our method.

D. Environment Partitioning

We evaluate our approach for environment partitioning (Sec. III-B) by comparing it against the ground-truth intersection points of the underlying map. Throughout exploring the environment, the recall is normalized using the point cloud of the road surface obtained thus far. Our proposed lane graph-based method (LG) is compared against a morphological
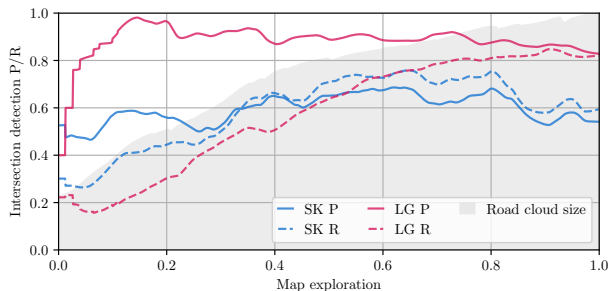
image skeletonization baseline (SK) that uses medial axes of the bird's-eye-view projected point cloud of the road surface. Kernelized smoothing and dilation followed by thresholding the obtained bird's-eye-view image helps in filtering false positive points and noise. In order to further increase precision, the SK baseline includes clustering culmination of intersection points in local areas that originate from artifacts in the skeleton graph. We report the precision and recall values across ten exploration runs on *town02* in Fig. 7. We observe that our approach (LG) achieves at least 20 % greater precision while showing comparable or exceeding recall scores. As our approach relies on observed vehicle trajectories, we attribute the lower initial recall of the LG method to a small number of initially seen trajectories while the point cloud-based baseline already processes a larger extent of the surroundings at this stage. Nonetheless, we observe that the SK baseline yields vastly different partitioning solutions throughout exploration as it is not robust to artifacts such as occlusions due to vehicles or sparse LiDAR readings of distant road surfaces. We believe that a conservative, high-precision classifier is beneficial as over-segmentation increases the number of roads and intersections unnecessarily. Further explanations are provided in suppl. material Sec. S.4. Additionally, we observe that simply extracting intersections from the pose graph produces low recalls as every path has to be traversed by the agents instead of relying on more descriptive observations.

V. Conclusion

In this work, we introduced CURB-SG as a novel approach to building large-scale hierarchical dynamic 3D urban scene graphs from multi-agent observations. We furthermore demonstrated how our collaborative SLAM approach facilitates frequent map updates and rapid exploration while scaling to large environments. To foster further research in this direction, we made our code publicly available. In future work, we will address the reliance on simulated panoptic labels and known initial poses of the agents. Orthogonal to that, follow-up work could address a decentralized variant that operates under real-time constraints. Furthermore, we plan to include pedestrian information as well as additional topological elements such as road boundaries.

REFERENCES

[1] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Conf. on Robot Learning*, 2018.

[2] D. Cattaneo, D. G. Sorrenti, and A. Valada, "CMRNet++: Map and camera agnostic monocular visual localization in LiDAR maps," *Int. Conf. on Robotics and Automation Workshop on Emerging Learning and Alg. Methods for Data Association in Robotics*, 2020.

[3] A. Diaz-Diaz, M. Ocaña, A. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa, "HD maps: Exploiting opendrive potential for path planning and map monitoring," in *IEEE Intelligent Vehicles Symposium*, 2022, pp. 1211–1217.

[4] R. Trumpp, M. Büchner, A. Valada, and M. Caccamo, "Efficient learning of urban driving policies using bird's-eye-view state representations," *Int. Conf. on Intelligent Transportation Systems*, 2023.

[5] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *Int. Conf. on Intelligent Transportation Systems*, 2018.

[6] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based semantic SLAM," in *Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4530–4537.

[7] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement," *Int. Journal of Adv. Rob. Systems*, vol. 16, no. 2, 2019.

[8] K. Wong, Y. Gu, and S. Kamijo, "Mapping for autonomous driving: Opportunities and challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 91–106, 2020.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *Int. Conf. on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[10] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-Graphs+: Real-time localization and mapping leveraging hierarchical representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4927–4934, 2023.

[11] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Robotics: Science and Systems*, 2022.

[12] I. Armeni, Z.-Y. He, A. Zamir, J. Gwak, J. Malik, M. Fischer, and S. Savarese, "3D scene graph: A structure for unified semantics, 3D space, and camera," in *Int. Conf. on Computer Vision*, 2019, pp. 5663–5672.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conf. on Robot Learning*, 2017.

[14] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, 2014.

[15] J. Arce, N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, "PADLoC: LiDAR-based deep loop closure detection and registration using panoptic attention," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1319–1326, 2023.

[16] D. Zou, P. Tan, and W. Yu, "Collaborative visual SLAM for multiple agents: A brief survey," *Virtual Reality and Intelligent Hardware*, vol. 1, no. 5, pp. 461–482, 2019.

[17] L. Riazuelo, J. Civera, and J. Montiel, "C^{2}TAM: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[18] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM — collaborative visual-inertial SLAM," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2762–2769, 2018.

[19] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "COVINS: Visual-inertial SLAM for centralized collaboration," in *IEEE Int. Symp. on Mixed and Augmented Reality Adjunct*, 2021, pp. 171–176.

[20] Y. Chang, K. Ebadi, C. E. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, *et al.*, "LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9175–9182, 2022.

[21] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *arXiv preprint arXiv:2301.06230*, 2023.

[22] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. on Robotics*, vol. 38, no. 4, pp. 2022–2038, 2022.

[23] Y. Huang, T. Shan, F. Chen, and B. Englot, "DiSCo-SLAM: Distributed scan context-enabled multi-robot LiDAR SLAM with two-stage global-local graph optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1150–1157, 2022.

[24] G. Kim and A. Kim, "Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 4802–4809.

[25] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[26] C. Lang, A. Braun, and A. Valada, "Robust object detection using knowledge graph embeddings," in *DAGM German Conference on Pattern Recognition*, 2022, pp. 445–461.

[27] M. Büchner, J. Zürn, I.-G. Todoran, A. Valada, and W. Burgard, "Learning and aggregating lane graphs for urban automated driving," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 13 415–13 424.

[28] Liao, Bencheng and Chen, Shaoyu and Wang, Xinggang and Cheng, Tianheng, and Zhang, Qian and Liu, Wenyu and Huang, Chang, "MapTR: Structured modeling and learning for online vectorized HD map construction," in *Int. Conf. on Learning Representations*, 2023.

[29] S. He and H. Balakrishnan, "Lane-level street map extraction from aerial imagery," in *IEEE Winter Conference on Applications of Computer Vision*, January 2022, pp. 2080–2089.

[30] N. Gosala, K. Petek, P. L. Drews-Jr, W. Burgard, and A. Valada, "Skyeye: Self-supervised bird's-eye-view semantic mapping using monocular frontal view images," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 14 901–14 910.

[31] R. Mohan and A. Valada, "Amodal panoptic segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 21 023–21 032.

[32] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Int. Journal of Computer Vision*, vol. 123, pp. 32–73, 2017.

[33] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *Robotics: Science and Systems*, 2020.

[34] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, "3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE Trans. on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2020.

[35] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Inferring and executing programs for visual reasoning," in *Int. Conf. on Computer Vision*, 2017, pp. 2989–2998.

[36] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[37] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, June 2021, pp. 7515–7525.

[38] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "EfficientLPS: Efficient LiDAR panoptic segmentation," *IEEE Trans. on Robotics*, vol. 38, no. 3, pp. 1894–1914, 2022.

[39] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Int. Conf. on Robotics and Automation*, 2021, pp. 11 054–11 059.

[40] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *Int. Conf. on Robotics and Automation*, 2011, pp. 3607–3613.

[41] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[42] G. Jozkow, P. Wieczorek, M. Karpina, A. Walicka, and A. Borkowski, "Performance evaluation of sUAS equipped with Velodyne HDL-32E LiDAR sensor," *The Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 171–177, 2017.

[43] A. Van, D. Lindenbaum, and T. M. Bacastow, "SpaceNet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.
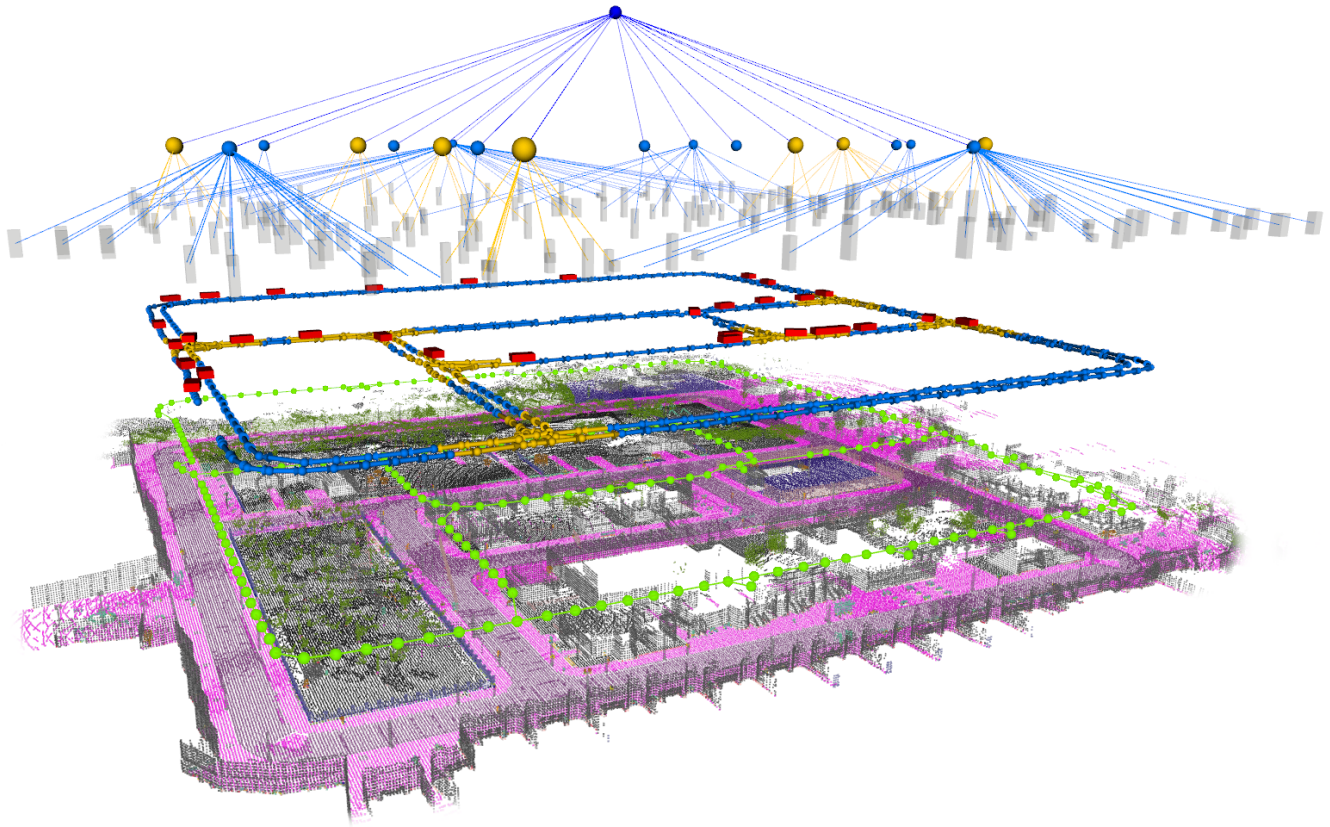
Collaborative Dynamic 3D Scene Graphs for Automated Driving

- Supplementary Material -

Elias Greve[1*], Martin Büchner[1*], Niclas Vödisch[1*], Wolfram Burgard[2], and Abhinav Valada[1]

In this supplementary material, we provide additional information on the implementation details of our proposed approach CURB-SG as well as a number of insights regarding our experimental evaluation. This ranges from our semantic voxel grid filtering and collaborative SLAM to our lane graph-based environment partitioning method. For convenience, we follow the structure of the main manuscript.

S.1. IMPLEMENTATION DETAILS

Throughout our evaluation, we make use of four different CARLA map environments as shown in Fig. S.1 that contain classical two-way roads (*town01, town02, town07*), roundabouts (*town07*) as well as bidirectional multi-lane roads (*town10*). We spawn 30 to 40 additional non-agent vehicles to simulate traffic behavior. The navigation of all agents and passive vehicles is random and for the agent-only evaluations,

* Equal contribution.
[1] Department of Computer Science, University of Freiburg, Germany.
[2] Department of Eng., University of Technology Nuremberg, Germany.

e.g. localization and map exploration, all the traffic lights are constantly green to avoid unnecessary waiting time for the agents.

We simulate a semantic $360°$ LiDAR sensor using the CARLA Python API with the following parameters: 32 channels, $2.8\,\mathrm{m}$ above ground surface, $10\,\mathrm{m}$ minimum distance, $10\,\mathrm{Hz}$ rotation frequency, 56,000 points per second, azimuth range of $[-30°, 10°]$. Our setup runs ROS Noetic and CARLA 0.9.13 on an Intel i5-6500 @ 4 x $3.2\,\mathrm{GHz}$ and an NVIDIA TITAN X GPU. The implementation uses both C++ (g++ 9.4.0) and Python (3.8).

S.2. SEMANTIC VOXEL GRID FILTERING

As briefly described in Sec. III-A, we adapt the original implementation of HDL Graph SLAM [7] regarding the prefiltering of point clouds. Instead of using a class-agnostic voxel grid filter, our semantic class-respective filter uses different grid sizes for different semantic classes. Therefore, smaller objects such as landmarks are represented using a higher number of points while typical "stuff" classes such

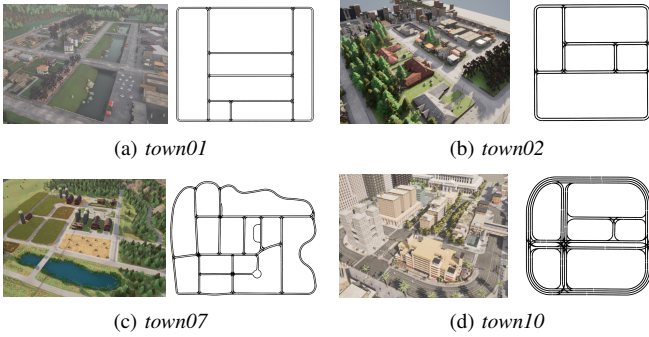(a) *town01* (b) *town02*

(c) *town07* (d) *town10*

Fig. S.1.   We use four different environments from the CARLA simulator [13].
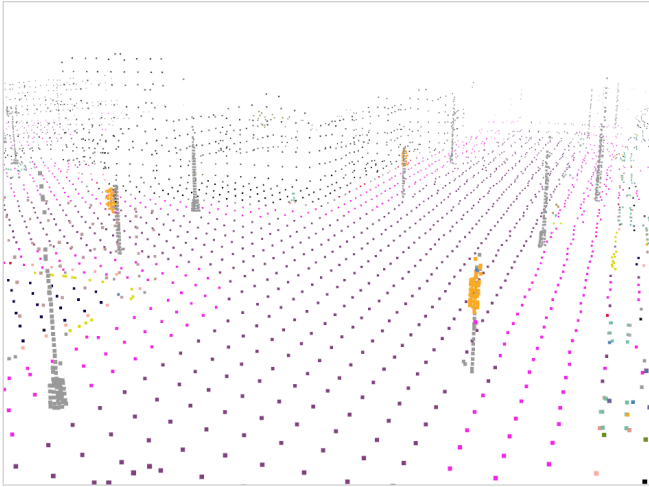

Fig. S.2.   Point cloud visualization of the obtained panoptic point cloud after semantics-based voxel grid filtering. As shown, typically small classes such as traffic signs and poles show a finer resolution while, e.g., the *road* class is filtered with a coarser grid.

TABLE S.1

SEMANTIC VOXEL GRID RESOLUTION.

Resolution [m]	Semantic class
0.15 (fine)	*Fence, Pole, TrafficSign, GuardRail, TrafficLight, Static, Dyn.*
0.5 (coarse)	*Unlabeled, Building, Other, Pedestrian, Road, SideWalk, Vegetation, Vehicles, Wall, Sky, Ground, Bridge, RailTrack, Water, Terrain*

All semantic classes given by CARLA and the respective voxel grid resolution utilized throughout our experiments.
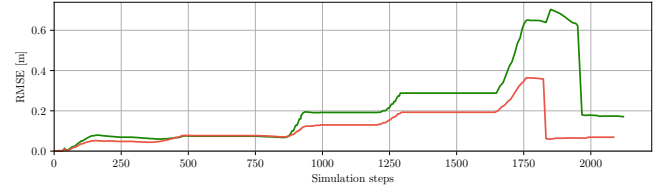
(a) RMSE of the agent localization of two runs with different filter approaches.

(b) RMSE of the traffic signs of two runs with different filter approaches.

Semantic voxel grid   General voxel grid

Fig. S.3.   Comparison of a general voxel grid filter to the semantic voxel grid filter.

as *road* use fewer points, as shown in Fig. S.2. The class-respective voxel grid resolutions are given in Tab. S.1.

This approach not only bears the potential to decrease the size of the stored point clouds but also leads to increased localization accuracy. As depicted in Fig. S.3, we observe smaller localization errors regarding both the agents as well as, e.g., traffic signs compared to the generic voxel grid filter. The shown data was collected by letting the agent go counterclockwise along the outside road of *town02* with a 15 m segment overlap at the end. Since a loop closure is detected at the end of the run, we observe a sudden decrease in the root mean squared error. After 2000 simulation steps, this induces a $\sim 50\%$ improvement using the semantic compared to the standard voxel grid filter.

S.3. COLLABORATIVE SLAM

Complementary to the findings in Fig. 4, we provide the exploration performance for one, two, and three agents on *town01* and *town10*. Similarly, we observe that three agents show the smallest time needed for 85% map exploration. However, as shown for *town01*, the effect of a third agent may potentially vanish on certain maps. Nonetheless, in an
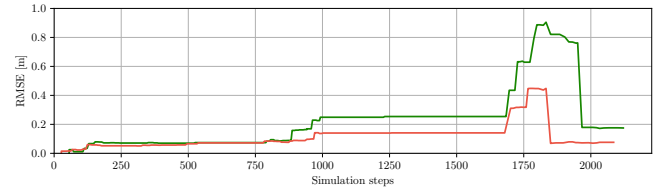
open-world setting, we believe that more agents lead to faster and more accurate mapping results as already shown in Tab. I.

S.4. 3D URBAN SCENE GRAPHS

A. Environment Partitioning

In addition to our findings in Sec. IV-D, we provide further qualitative and quantitative findings. Different from Fig. 7, in Fig. S.6, we show the intersection detection precision (P) and recall (R) for a single exploration run instead of the average across 10 runs. While the precision of the lane graph-based method (LG) stays constantly at 1.0 the recall generally increases throughout exploration. Minor decreases in the recall score are compensated immediately. In contrast, the image-based skeletonization method (SK) shows highly volatile precision and recall curves that do not improve while greater extents of the environment are explored. In general, this leads to a vast number of different segmentation solutions while exploring, which is generally not suited for robust mapping.

In Fig. S.4, we portray the qualitative output of CURB-SG on *town02*. A directed graph colored in black represents the lane graph, whereas a green graph slightly above the semantic point cloud denotes the obtained pose graph that is subject to (inter-agent) loop closures. Yellow-colored nodes above the lane graph denote intersections while blue nodes represent
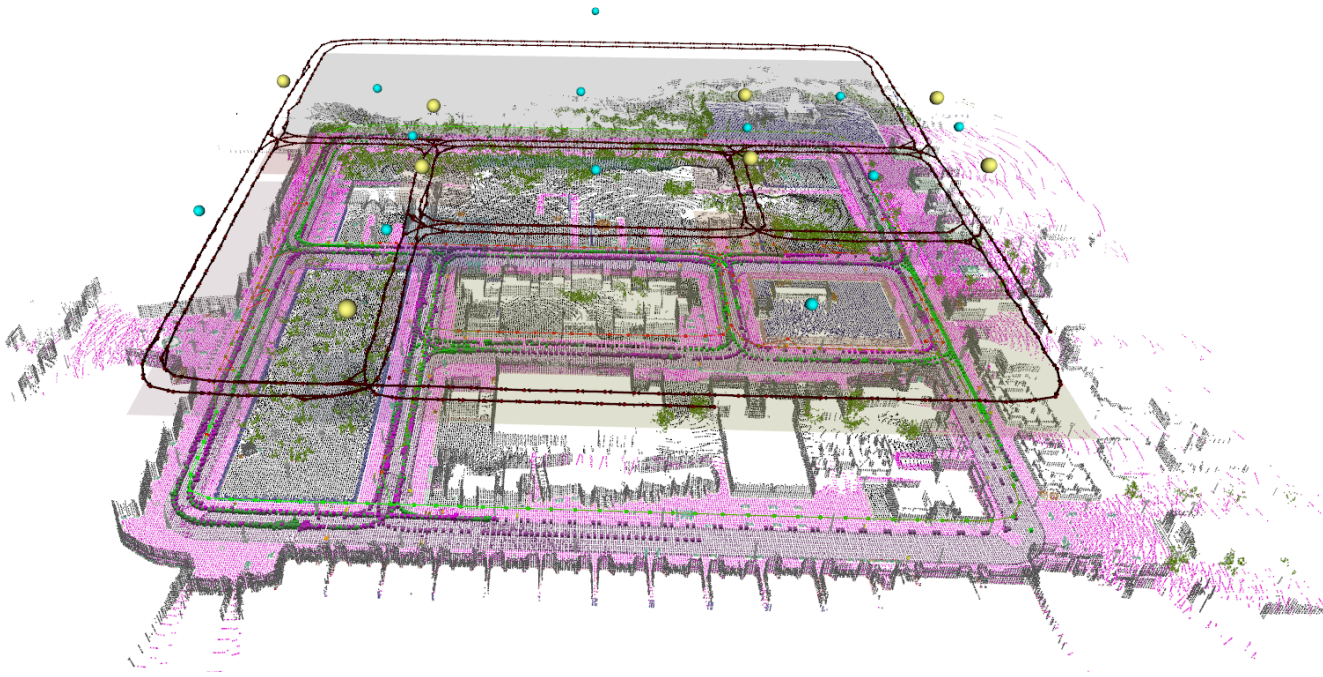
Fig. S.4. Qualitative visualization of the intersection detection quality. Yellow nodes represent intersections, whereas blue nodes represent non-intersecting road areas. As shown, these areas can be predicted with high precision even without a complete lane graph.
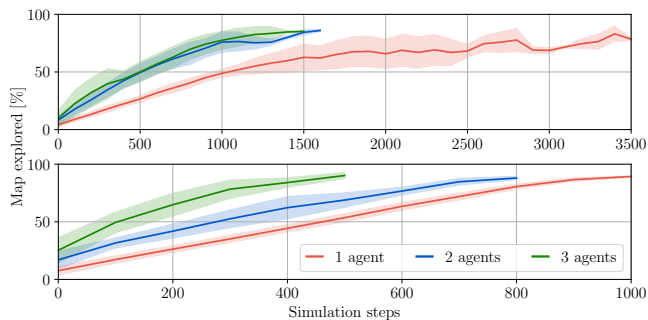

Fig. S.5. The mapping progress in *town01* (top) and *town10* (bottom) for one, two, and three agents. Our collaborative SLAM method benefits from receiving inputs from multiple agents. Although additional agents might not further increase the mapping speed, they will continue contributing towards frequent map updates and refining the lane graph.
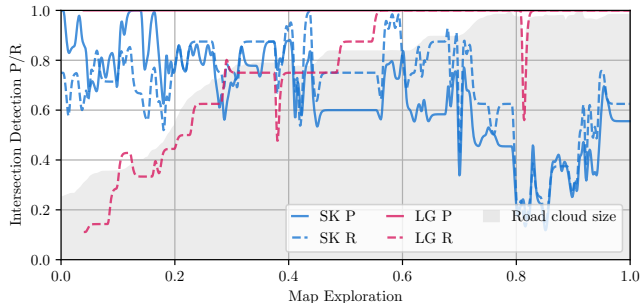

Fig. S.6. Intersection detection quality of a single run on map *town02*. Precision (P, solid) and recall (R, dashed) of both our lane graph-based method (LG) as well as the skeletonization baseline (SK) are plotted while the map sees exploration.

non-intersecting road segments. Although the produced lane graph shows slight inaccuracies and is not fully complete, we are able to segment areas under both high precision and recall.