

Elastic and Efficient LiDAR Reconstruction for Large-Scale Exploration Tasks

Yiduo Wang¹, Nils Funk², Milad Ramezani¹, Sotiris Papatheodorou², Marija Popović², Marco Camurri¹,
Stefan Leutenegger² and Maurice Fallon¹

Abstract—We present an efficient, elastic 3D LiDAR reconstruction framework which can reconstruct up to maximum LiDAR ranges (60 m) at multiple frames per second, thus enabling robot exploration in large-scale environments. Our approach only requires a CPU. We focus on three main challenges of large-scale reconstruction: integration of long-range LiDAR scans at high frequency, the capacity to deform the reconstruction after loop closures are detected, and scalability for long-duration exploration. Our system extends upon a state-of-the-art efficient RGB-D volumetric reconstruction technique, called *supereight*, to support LiDAR scans and a newly developed submapping technique to allow for dynamic correction of the 3D reconstruction. We then introduce a novel pose graph sparsification and submap fusion feature to make our system more scalable for large environments. We evaluate the performance using a published dataset captured by a handheld mapping device scanning a set of buildings, and with a mobile robot exploring an underground room network. Experimental results demonstrate that our system can reconstruct at 3 Hz with 60 m sensor range and ~ 5 cm resolution, while state-of-the-art approaches can only reconstruct to 25 cm resolution or 20 m range at the same frequency.

I. INTRODUCTION

Dense surface reconstruction is an active research topic. Being able to recover rich geometric information in real time is important for applications such as active mapping [1], [2], obstacle avoidance [3] and industrial inspection [4], [5]. Lower cost and denser LiDAR sensors have come to the market thanks to the focus on self-driving car research. However, large scale exploration and reconstruction still remain challenging problems.

A major challenge in reconstruction is global consistency because the accumulation of some degree of odometry error is unavoidable during large-scale exploration — even for approaches such as LOAM [6]. Error increases with distance travelled and is typically corrected by loop closures in the context of SLAM. Take pose-graph SLAM as an example. Upon the detection of a loop closure, SLAM systems introduce a new constraint between the head and the tail of the loop, optimise the pose graph and propagate pose correction back through the trajectory. These corrections

This research is supported by the ESPRC ORCA Robotics Hub (EP/R026173/1). M. Fallon is supported by a Royal Society University Research Fellowship and S. Papatheodorou by the President's PhD Scholarship.

¹ These authors are with the Oxford Robotics Institute, University of Oxford, UK. {ywang, milad, mcamurri, mfallon}@robots.ox.ac.uk

² These authors are with the Smart Robotics Lab, Department of Computing, Imperial College London, UK. {nils.funk13, s.papatheodorou18, mpopovil, s.leutenegger}@ic.ac.uk

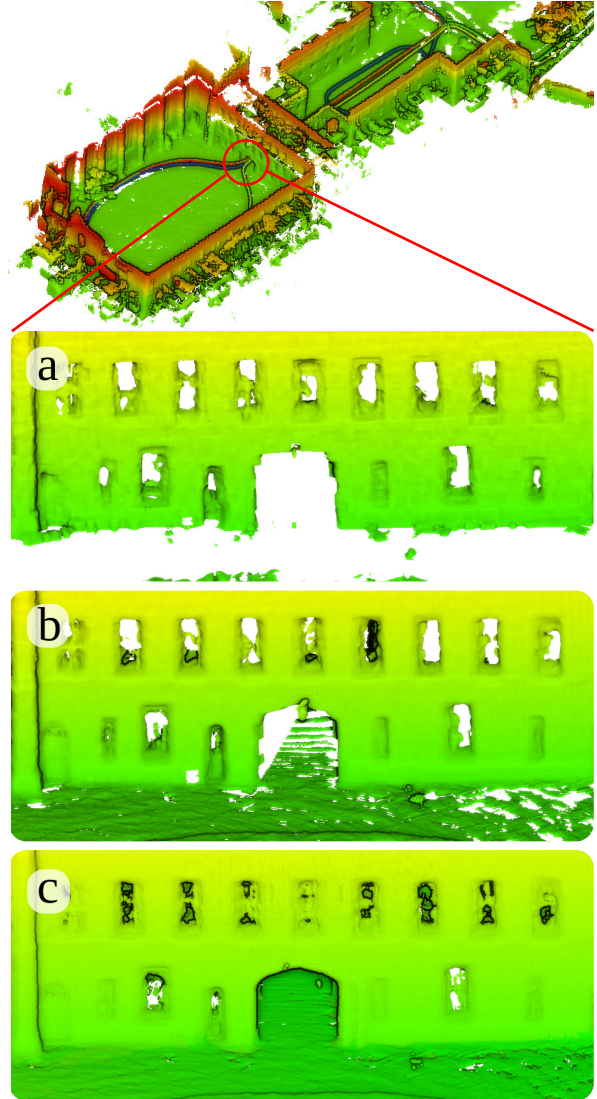


Fig. 1: Exploration trajectory and 3D reconstruction result from our elastic *supereight* multi-resolution TSDF pipeline on Newer College Dataset. The close-ups focus on the narrow tunnel on the opposite side of the Quad area from experiment's start. (a - the first submap 40 m away; b - before going through the tunnel; c - revisiting after a large loop closure.)

can also be applied to the map representation used within SLAM systems, which are typically point clouds for LiDAR-based SLAM. However, if a dense reconstruction (such as a surface mesh) is built on-the-fly by an exploring robot,

this reconstruction will be rigid, making it impossible to incorporate the effect of loop closures.

Another challenge is finding a good trade-off between the resolution/scale of the reconstruction, and the speed/efficiency of the system. A precise representation of occupancy is important for robot path planning, especially when planning paths through tunnels and door ways such as in Fig. 1. *Supereight* [7], a state-of-the-art reconstruction framework for RGB-D cameras, maintains multiple reconstructions with an adaptive resolution at different scan ranges. This approach uses a high resolution reconstruction in close proximity for path planning and lower resolution at longer ranges for fast reconstruction. In this work, we greatly expand *supereight* to incorporate 3D multi-beam LiDAR scans. We then experimentally demonstrate that our approach is more efficient than other state-of-the-art pipelines at high resolutions.

Finally, we also implement a novel system to sparsify the SLAM pose graph and reconstruction, inspired by large-scale systems such as the *Atlas* SLAM framework [8]. Reconstructions of the same physical space are fused together to avoid redundant mapping.

The contributions of our research are the following:

- An elastic 3D reconstruction system that can support corrections to its underlying shape, e.g. from loop closures.
- A pose graph sparsification and submap fusion strategy that makes the reconstruction’s memory usage grow proportionally with the size of the environment rather than the duration of exploration.
- Incorporation of LiDAR into a state-of-the-art reconstruction framework which achieves multi-fps (3 Hz) full range (60m) LiDAR scan integration with high resolution (~ 5 cm) to enable high precision motion planning and long range autonomy. To the best of our knowledge, this is the first system that achieves this level of performance.
- Evaluation of the system using real-world datasets in large-scale environments against state-of-the-art methods such as Octomap and Voxgraph.

The remainder of this paper is organised as follows. In Section II we discuss the related work. Section III focuses on our reconstruction method and Section IV explains how we achieve elasticity. Experimental results are presented in Section V. Section VI discusses conclusions and future work.

II. RELATED WORK

Dense SLAM and mapping systems are very active areas of research, and a wide variety of systems have been designed with different use cases. Our proposed system mainly focuses on two aspects, namely large-scale dense reconstruction, and submaps and elasticity. In this section, we will give a brief review of the most relevant systems.

a) Dense Reconstruction: There are a variety of representations for 3D environments. Newcombe *et al.* [9] used a global, densely-allocated Truncated Signed Distance Function (TSDF) volume to achieve reconstructions with ground-

breaking details. Alternatively, the approach of Whelan *et al.* [10] uses surfels as its primary surface representation. Both systems utilise a GPU to integrate inputs from RGB-D cameras, which typically have a maximum accurate sensing range of only 3m as compared to as much as 100m for terrestrial LiDAR. Recent work by Park *et al.* [11] revised the dense surfel model and proposed novel representation and matching methods for dense LiDAR SLAM. Another technique to improve the scalability of reconstruction is dynamic allocation via data structures such as octrees [12], [13] and hash-tables [14], [15]. For instance, Niessner *et al.* [16] employed a TSDF for large-scale mapping by exploiting the sparsity of environment via a technique called Hashing Voxel Grid (HVG). Tanner *et al.* [17] also utilised HVG for efficient large-scale TSDF reconstruction over kilometers. Their pipeline BOR²G further incorporates multiple types of sensor inputs, including long-range LiDAR.

While surface-based reconstructions produce accurate representations for structures and occupied space, they do not distinguish between free and unknown space and are difficult to use for robotics operations such as path planning. As a result, systems designed for navigation and motion planning prefer to use occupancy grids for dense reconstruction to explicitly represent free and unknown space [18]–[21].

Vespa *et al.* [7] presented *supereight*, a volumetric SLAM system that uses an efficient octree structure to store either TSDF or Occupancy information to support mapping or planning respectively. *Supereight* represents the sensed space with a grid of voxels, and can integrate and render RGB-D measurements at various levels of detail depending on the distance between sensor and surface to improve efficiency. In this work, we have extended the algorithm to support the much longer ranges from 3D LiDAR sensors, while also utilising the feature of adaptive resolution and both TSDF and Occupancy pipelines for reconstruction and path planning, respectively.

b) Submaps and Elasticity: Online mapping systems need to track the sensor pose at each frame before integrating scans into a reconstruction. Odometry error accumulated through incremental tracking methods, such as frame-to-frame or frame-to-model, can therefore lead to distortion in the map. Dense SLAM systems such as KineticFusion [9] and ElasticFusion [10] use map-centric approaches to tightly couple their reconstructions with the SLAM trajectory, and bend the mapped environment upon loop closure. De Gregorio and Di Stefano [18] proposed occupancy-based methods to erode and re-integrate past scans individually from the existing reconstruction upon pose graph optimisation. These methods, however, suffer from poor scalability in large-scale online operations.

Another technique to improve global consistency is to represent the full 3D reconstruction using a collection of submaps of limited extent. This technique originated in SLAM research such as the *Atlas* framework by Bosse *et al.* [8] and DenseSLAM by Nieto *et al.* [22]. Both systems maintain an interconnected collection of local submaps instead of a single global map, which sparsifies the environ-

ment even in the case of dense reconstruction. Additionally, *Atlas* reuses existing maps instead of spawning a new submap upon loop closure, hence allowing the global map to scale with the size of the explored environment instead of the exploration length.

Ho *et al.* [19] and Reijgwart *et al.* [23] both exploited submaps to achieve elasticity in dense reconstruction for the purpose of motion planning. Their reconstructions are based on OctoMap [24] and Voxblox [3], respectively. Upon loop closure, submaps in both systems can be moved around to keep global consistency. However, these systems both have some limitations. For motion planning, the system of Ho *et al.* [19] requires raycasting into every submap for occupancy information, significantly increasing the complexity when there are many submaps. The authors improved their method in [20] where submaps are merged together into a global map for faster voxel query. Global map update is only triggered when a submap’s pose changes significantly. In a large-scale environment with a long range-sensor, however, maintaining dense reconstructions of both submaps and a global map in memory is very inefficient.

Our approach is most directly motivated by the work of Reijgwart *et al.* [23]. Their map-centric dense SLAM pipeline, Voxgraph, constructs TSDF submaps and generates correspondence-free constraints among them for global consistency. It can also be employed for robotic tasks like path planning by computing a Euclidean Signed Distance Field (ESDF) using the TSDF. Voxgraph incorporates both LiDAR scans and RGB-D measurements, and was demonstrated to be sufficiently lightweight to run onboard a Micro Aerial Vehicle. The main drawback of this system is the dense data structure of hash-tables. TSDF integration and ESDF computation are all carried out at the finest resolution. The typical parameter setup the authors used for a LiDAR exploration task was short range (16 m) with coarse resolution (20 cm). With such parameters, it would be impossible to plan at full LiDAR ranges.

III. SUPEREIGHT WITH LiDAR

In this section, we describe the core reconstruction component of our system, *supereight* [7], and the significant improvements needed to support long range LiDAR sensing. *Supereight* is a volumetric, octree-based SLAM pipeline that uses Morton codes to achieve efficient spatial octree traversals. Rather than storing voxels at the finest level of the octree, *supereight* stores blocks which aggregate $8 \times 8 \times 8$ voxels as the leaves of the tree. This results in fewer memory allocations and improved cache locality during updates, both of which improve performance. It is also capable of integrating data at different octree levels, thus reducing aliasing artifacts and increasing performance.

In this work, the mapping component of *supereight* is connected to a LiDAR SLAM system [25]. We expand both *supereight*’s multi-resolution TSDF (*MultiresTSDF*) [7] and multi-resolution occupancy (*MultiresOFusion*) [26] pipelines to incorporate LiDAR inputs, by adding a spherical projection LiDAR sensor model. We also create local submaps in

this system to replace the single global map in the original pipeline. This is further explained in Sec. IV.

A. Multi-resolution

Supereight can update the octree at various levels, depending on the effective resolution of the sensor. Long range measurements can cover large free space with little information. Instead of updating individual voxels, *supereight* updates cubes consisting of several voxels. The benefit of this approach is a reduction in the number of octree updates compared to always updating at the voxel level, resulting in reduced integration time [7]. This performance increase is especially important in the case of LiDAR sensors where a single scan may contain measurements ranging from a few meters to 60 m away.

In this work, we update *supereight*’s integration level selection method for a particular depth measurement to make it suitable for LiDAR sensors, with the aim of reducing aliasing artefacts, as well as increasing speed and decreasing memory consumption at large distances. Given a distance measurement d_r along the ray \mathbf{r} , we consider the minimum angle between two adjacent LiDAR scan rays, in terms of both azimuth and elevation. With this angle, we create a circular cone around \mathbf{r} , which we can use to compute the radius of the largest sphere that would fit inside the cone at the distance d_r . We then select the integration level whose volume’s diagonal is the closest to the sphere’s diameter, up to 3 levels above the single-voxel leaf level. A single integration level is always chosen for an entire leaf block. Thus, measurements can be integrated into volumes at adaptively selected resolutions ranging from $1 \times 1 \times 1$ to $8 \times 8 \times 8$ voxels within each block.

We use the propagation strategies described in [7] and [26] for MultiresTSDF and MultiresTSDF respectively to keep the hierarchy consistent between different integration levels. Additionally, in MultiresTSDF the maximum occupancy and observed state at the finest integration level are up-propagated to each parent level, up to the octree’s root, to provide fast occupancy queries at different levels (and to accelerate the raycasting stage, if needed in tracking). After each integration, the octree is pruned to remove redundant details: while MultiresTSDF allocates and updates the volume densely within a band around the surface, MultiresOFusion additionally keeps track of free space at the coarsest possible scale while preserving details about unknown areas of the map due to occlusion, holes in the depth image or frustum boundaries.

B. LiDAR Integration

LiDAR integration refers to the process of creating a new reconstruction or updating an existing one based on a new LiDAR scan. The LiDAR used in our experiments is an Ouster OS1-64. It produces organised dense point clouds of 64×1024 points at 10 Hz (*i.e.* 655k points/s), with a vertical Field of View (FoV) of 33.2° and a horizontal FoV of 360° . Scans are converted from point clouds to spherical range images to facilitate their inclusion into *supereight*.

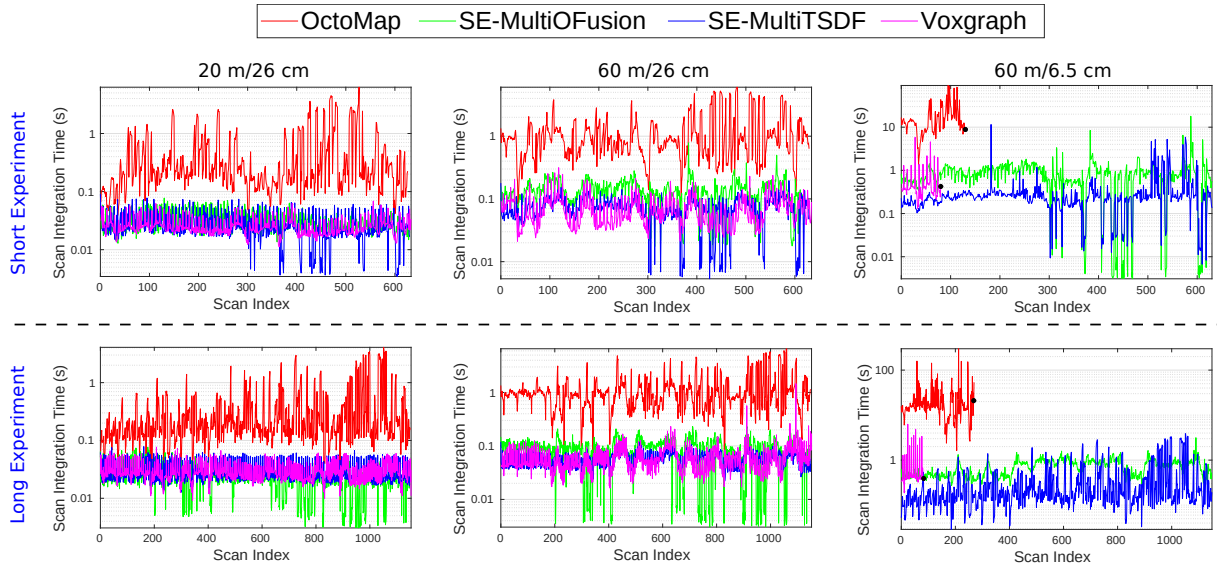


Fig. 2: Integration time per LiDAR scan of different reconstruction systems in large-scale exploration experiments. Our goal is to achieve high resolution at maximum sensor range (right column).

The MultiresTSDF pipeline stores the occupancy probability in log-odds form which results in free, unknown and occupied voxels having negative, zero and positive log-odds values, respectively. Occupancy update follows the convention of adding a new log-odds measurement [24], [27].

Given a distance measurement d_r along a ray \mathbf{r} , we assume its standard deviation is $\sigma(d_r) = \max(\sigma_{\min}, k_\sigma d_r)$ where σ_{\min} and k_σ are constants that depend on the sensor characteristics. The log-odds occupancy probability $L(d)$ at a distance d along \mathbf{r} is computed in a manner inspired by [28] but using a piecewise linear function instead:

$$L(d) = \begin{cases} l_{\min} & \text{if } d \leq 3\sigma(d_r) \\ \frac{-l_{\min}}{3\sigma(d_r)}d & \text{if } 3\sigma(d_r) < d \leq \frac{k_\tau d_r}{2} \\ \frac{-l_{\min}}{3\sigma(d_r)}\frac{k_\tau d_r}{2} & \text{if } \frac{k_\tau d_r}{2} < d \leq k_\tau d_r \\ \text{no update} & \text{otherwise} \end{cases} \quad (1)$$

where l_{\min} is a negative constant denoting the minimum occupancy probability in log-odds and k_τ is a positive constant scaling factor used to control how much occupied space is created behind the measurement. The values used for these parameters in our experiments were $\sigma_{\min} = 1.5 \times$ voxel resolution, $k_\sigma = 0.1$, $l_{\min} = \log_2 \frac{0.03}{1-0.03}$ and $k_\tau = 0.1$.

The (alternative) integration process for the MultiresTSDF pipeline is described in [7]. In order to avoid artefacts that appear only in long-range LiDAR scans, we modify the pipeline so that the TSDF truncation bound adapts to the integration level instead of being constant. More specifically, the truncation bound is set to 8 times the edge length of the volume at the current integration level.

C. Runtime Performance

To evaluate our system's efficiency when integrating LiDAR scans, we tested it using the Newer College Dataset (NCD) [29]. This dataset was collected in a large-scale environment (approximately $135 \times 225 \text{ m}^2$) with a hand-held

multi-sensor equipped with an Ouster OS1-64 LiDAR and a RealSense D435i camera. It consists of two experiments with different durations, about 25 min (NCD short experiment) and 44 min (NCD long experiment).

The baseline algorithms we chose for comparison are OctoMap [24] and Voxgraph [23], to assess the respective Occupancy and TSDF pipelines. For these experiments, we fed one point cloud every 2 m travelled into the reconstruction systems. All reconstruction computations were performed on a laptop with an Intel® Xeon E3-1505M v6 CPU, 16 GB of RAM and 32 GB of swap memory.

We evaluated the computation time using three different sets of maximum scan range and voxel resolutions:

- 20 m max range with 26 cm resolution
- 60 m max range with 26 cm resolution
- 60 m max range with 6.5 cm resolution

Fig. 2 shows the integration time at the different range/resolution combinations for both the short and the long experiments (top and bottom rows, respectively). We focus on mapping at high resolution (6.5 cm) with maximum LiDAR range (60 m), which is presented in the right column of Fig. 2. In these experiments, OctoMap and Voxgraph both terminated early due to memory limit. Fig. 3 shows the memory consumption of each pipeline, as well as the growth of submaps in our proposed system. The memory usage of OctoMap and Voxgraph increases more quickly than both *supereight* pipelines, thus illustrating how the multi-resolution feature of *supereight* improves the memory efficiency of the reconstruction, allowing it to scale to larger environments.

Overall, OctoMap is the least efficient of the evaluated methods. With coarse resolutions, Voxgraph exhibits similar performance to *supereight*. However, at 6.5 cm resolution, the MultiresTSDF pipeline is faster than Voxgraph, while MultiresTSDF is on a par with Voxgraph.

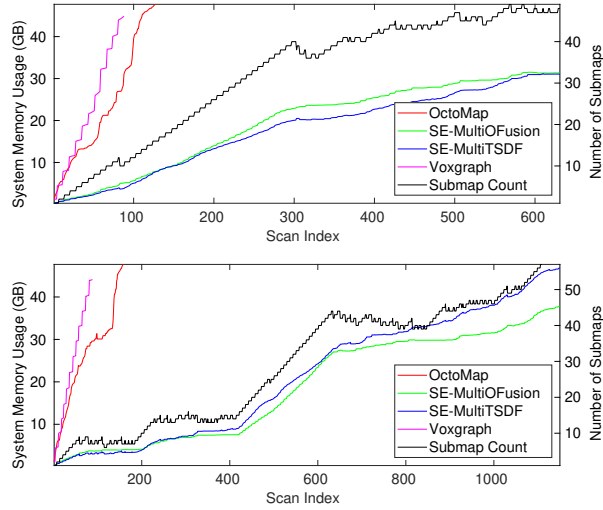


Fig. 3: Memory usage of each pipeline in the NCD short (top) and long (bottom) experiments with 60m range and 6.5cm resolution. The memory usage of our pipelines had a non-linear profile in the long experiment because of the submap fusion feature. (See Sec. IV-C for more details)

IV. SUBMAPS AND ELASTICITY

In this section, we detail the components of our elastic reconstruction pipeline, which is shown in Fig. 4.

Our pose graph SLAM system [25] fuses the sensors signals described in Sec. III-C to compute relative odometry which is locally consistent with drift rates in the order of 1m per 100m travelled. In this way we collect a sequence of point clouds which are registered to one another locally, as well as a corresponding relative pose estimate for the robot/device. As loop closures are determined we form a full pose graph. We call this a *Registered Cloud List*.

Fig. 5 illustrates our frame conventions. The Map frame $\{\mathcal{M}\}$ defines a global fixed frame of reference. The base frame of the device at time k is defined as $\{\mathcal{B}_k\}$. The SLAM system provides a pose graph with Q nodes $X_k, k \in \{0, \dots, Q\}$. Each node describes the estimated pose of the device expressed in the map frame ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{B}_k} \in \mathbf{SE}(3)$. The graph's topology consists of both odometry edges (*i.e.* connecting two consecutive nodes) and loop closure edges.

Each node of the graph is associated with a raw point cloud from the LiDAR. The point clouds have a fixed number of points $p \in \mathbb{R}^3$ expressed in the LiDAR frame $\{\mathcal{L}\}$. Given a node X_k and its associated point cloud C_k , the pose of the LiDAR ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{L}_k}$ can be computed as follows:

$${}^{\mathcal{M}}\mathbf{T}_{\mathcal{L}_k} = {}^{\mathcal{M}}\mathbf{T}_{\mathcal{B}_k} {}^{\mathcal{B}}\mathbf{T}_{\mathcal{L}} \quad (2)$$

where ${}^{\mathcal{B}}\mathbf{T}_{\mathcal{L}} \in \mathbf{SE}(3)$ is a static transform known by design.

The output of the reconstruction system consists of N submaps. Each submap $\mathcal{S}_i, i \in \{0 \dots N\}$ contains:

- The reconstruction as an occupancy map or TSDF
- The root pose of \mathcal{S}_i
- The node indices and scans used to construct \mathcal{S}_i

A submap's root pose defines the submap's transformation

with respect to the map frame ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{S}_i}$. A submap's root pose is fixed.

A. Graph Sparsification

The *graph sparsification* module processes the pose graph of the Registered Cloud List and groups graph nodes together into different submaps. The sparsified graph further guides *scan integration* (Sec. IV-B) and *submap fusion* (Sec. IV-C).

To perform sparsification, we first divide the pose graph edges into odometry and loop closure edges. Odometry edges represent constraints between consecutive pairs of nodes, while loop closure edges are the constraints between nodes that are distant in the graph but correspond to similar scans of revisited places.

If there are no loop closures, the grouping into submaps is based only on the odometry chain with a distance λ_{odom} . In this case, the first node $X_{i,0}$ of submap \mathcal{S}_i defines the submap's root pose ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{S}_i}$ with its corresponding LiDAR pose ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{L}_{i,0}}$.

For the subsequent nodes, we compute the distance travelled along the pose graph from the root pose. If a new node is within the distance threshold λ_{odom} , the associated LiDAR scan is integrated into \mathcal{S}_i according to Sec. IV-B. When the new node exceeds the distance threshold, a new submap \mathcal{S}_{i+1} is spawned with that node. This is based on the assumption that the odometry drift is proportional to distance travelled.

Upon loop closure, we cluster together nodes that are within a threshold λ_{cluster} around the pair of nodes that form the closure. Fig. 6a presents an example of clustering. In this example, \mathcal{L}_2 and \mathcal{L}_7 are connected by a loop closure edge. We then compute the distances along pose graph edges from every surrounding node to this loop closure pair. In the case of \mathcal{L}_9 , its distance is computed as:

$$\begin{aligned} d_{\mathcal{L}_7, \mathcal{L}_9} &= d_{\mathcal{L}_7, \mathcal{L}_8} + d_{\mathcal{L}_8, \mathcal{L}_9} \\ &= \|{}^{\mathcal{M}}\mathbf{t}_{\mathcal{L}_7} - {}^{\mathcal{M}}\mathbf{t}_{\mathcal{L}_8}\| + \|{}^{\mathcal{M}}\mathbf{t}_{\mathcal{L}_8} - {}^{\mathcal{M}}\mathbf{t}_{\mathcal{L}_9}\| \end{aligned} \quad (3)$$

and because $d_{\mathcal{L}_7, \mathcal{L}_9} < \lambda_{\text{cluster}}$, \mathcal{L}_9 is included in this cluster. Loop closure clusters are then used to guide *submap fusion* in Sec. IV-C.

B. Scan Integration

After sparsification is performed, a point cloud C_k at node X_k is integrated into the submap \mathcal{S}_i . We first compute the relative pose between \mathcal{L}_k and \mathcal{S}_i :

$${}_{\mathcal{S}_i}\mathbf{T}_{\mathcal{L}_k} = {}^{\mathcal{M}}\mathbf{T}_{\mathcal{S}_i}^{-1} {}^{\mathcal{M}}\mathbf{T}_{\mathcal{B}_k} {}^{\mathcal{B}}\mathbf{T}_{\mathcal{L}} \quad (4)$$

When X_k is the first node of a submap, its cloud C_k creates the initial reconstruction of \mathcal{S}_i and ${}^{\mathcal{M}}\mathbf{T}_{\mathcal{S}_i} = {}^{\mathcal{M}}\mathbf{T}_{\mathcal{L}_k}$.

Using ${}_{\mathcal{S}_i}\mathbf{T}_{\mathcal{L}_k}$, we transform every point $p \in C_k$ from the LiDAR sensor frame $\{\mathcal{L}_k\}$ to the submap frame $\{\mathcal{S}_i\}$. We then apply the method presented in Sec. III-B to update the submap reconstruction accordingly.

C. Submap Fusion

Submap fusion merges the submaps where a loop closure is detected (either geometrically or visually). For each loop closure cluster described in Sec. IV-A, we search through all existing submaps and find those that contain nodes from this

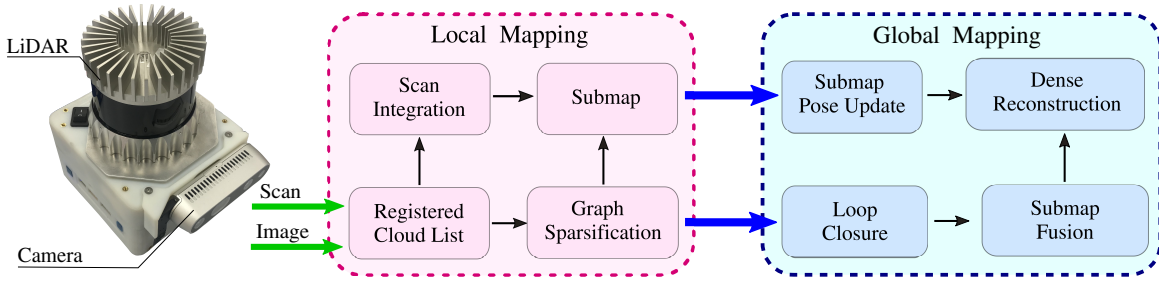


Fig. 4: An overview of the system. Registered Cloud List is the input to our system. Local mapping sparsifies the SLAM pose graph and integrates individual LiDAR scans into submaps. Global mapping updates poses of submaps upon loop closures and fuses overlapping submaps together. See Sec. IV for more details.

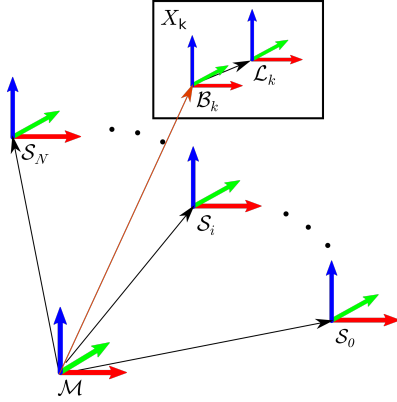


Fig. 5: The SLAM frame convention. The orange arrow refers to the input from a SLAM system. The black arrow refers to the transforms created inside our system.

cluster. These submaps are then fused together as illustrated in Fig. 6b.

To fuse the submaps S_j and S_i , we first need to transform every voxel of S_j with coordinates $v_j \in \mathbb{R}^3$ into the coordinate system of S_i to obtain v_i :

$$\begin{bmatrix} v_i \\ 1 \end{bmatrix} = s_i \mathbf{T}_{S_j} \begin{bmatrix} v_j \\ 1 \end{bmatrix} \quad (5)$$

$$s_i \mathbf{T}_{S_j} = {}^{\mathcal{M}}\mathbf{T}_{S_i}^{-1} {}^{\mathcal{M}}\mathbf{T}_{S_j}$$

If the voxel v_i falls out of the current scanned space in S_i , it will be newly allocated and assigned as v_j in S_j . Otherwise, the voxel data in v_j will be integrated into v_i following the model in Sec. III-B.

Submap fusion prevents new submaps from being spawned when the same space is revisited. Updating an existing submap is more memory efficient than creating two overlapping submaps. This has the advantage of making the reconstruction complexity grow proportionally with the amount of space explored rather than the duration of the exploration.

The memory usage of the long range (60m) high resolution (6.5cm) NCD experiments, as presented in Fig. 3, demonstrates such benefit. Fig. 3 also presents the growth of submaps in both experiments. Up to scan 400 in the NCD long experiment, the number of submaps has limited growth because the experiment stays within the Quad area and loops are closed. There after the device explored the wide open

Parkland area - with submap growth becoming linear. Ideally submap growth should have fully plateaued when revisiting the same area regularly (after scan 600). Improving submap reduction to achieve the plateauing is a future work.

D. Submap Pose Update

The *submap pose update* module ensures global consistency in our reconstruction. When loop closure occurs, the pose graph and poses of the SLAM system are updated.

The naive approach of updating all the submaps upon loop closure is computationally infeasible for real-time applications, as discussed by Sodhi *et al.* [20]. Instead, we define a criterion to determine whether a submap S_i needs to be corrected, such that a large-scale reconstruction can be selectively and efficiently updated.

Let ${}^{\mathcal{M}}\hat{\mathbf{T}}_{S_i}$ denote the updated transformation ${}^{\mathcal{M}}\mathbf{T}_{S_i}$ of S_i with respect to the map frame \mathcal{M} . We empirically determined translational and rotational thresholds which trigger a submap correction, respectively 10 cm or 2.5° . If the position/rotation change exceeds its threshold, the submap is corrected:

$$\|{}^{\mathcal{M}}\hat{\mathbf{t}}_{S_i} - {}^{\mathcal{M}}\mathbf{t}_{S_i}\| > \lambda_{\text{update}} \vee \|{}^{\mathcal{M}}\hat{\mathbf{R}}_{S_i}^{-1} {}^{\mathcal{M}}\mathbf{R}_{S_i}\| > \theta_{\text{update}} \quad (6)$$

Because we do not maintain a global map in our pipeline, this update need only correct the root poses of the submaps, with no additional global map fusion required.

V. EXPERIMENTS AND EVALUATION

In this section, we evaluate the global consistency of our online elastic reconstruction pipeline using MultiresTSDF and test the path planning application of MultiresOFusion. To assess the level of global consistency achieved via submap elasticity, we compared the MultiresTSDF map with the ground truth of NCD.

For path planning, we tested on a dataset of a mobile robot exploring an network of rooms in a underground mine, and used the MultiresTSDF pipeline to create a high resolution occupancy map so as to test suitability for path planning.

A. Reconstruction Accuracy

In Fig. 7, we present the result of the MultiresTSDF reconstruction on the NCD long experiment [29]. It is an exploration task spanning nearly 2.2 km in an approximately $135 \times 225 \text{ m}^2$ college campus. Given the scale, the 60 m

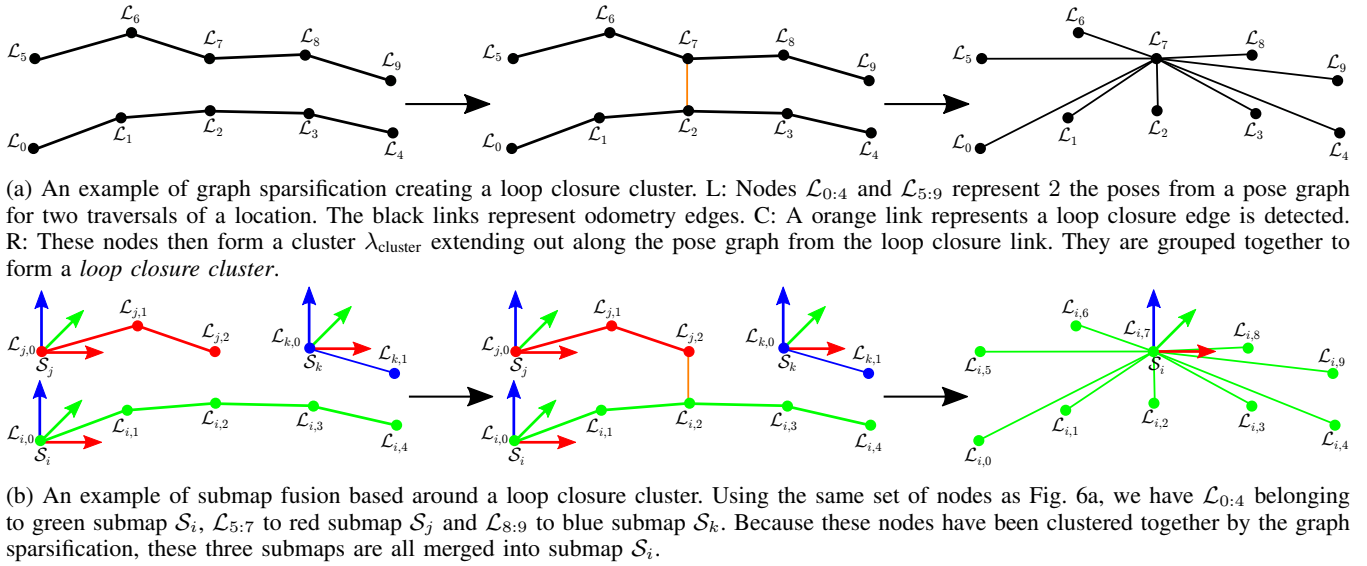


Fig. 6: An example of graph sparsification and submap fusion upon loop closure.

maximum sensor range and 6.5 cm voxel resolution is needed for this reconstruction.

To evaluate the map quality, we fused all the submaps generated at the end of the experiment into one global MultiresTSDF reconstruction and extracted all the vertices of the mesh to form a full point cloud of the explored environment. We then compared the extracted cloud with a ground truth map collected with a Leica BLK 360, a survey grade tripod-based laser scanner. The ground truth cloud was downsampled to 5 cm to match the resolution of our reconstruction. We used CloudCompare¹ to align the ground truth and the reconstructed point cloud and to compute the point-to-point distance error between them.

For about 90% of the points, the distance error is less than 50 cm. Additionally, the tunnel and the doorway connecting each section of the campus were reconstructed with high accuracy. This is demonstrated in detail in Fig. 1, where the finely detailed mesh of the tunnel enables tasks such as robot path planning.

B. Path Planning in Underground Network

To test the MultiresTSDF pipeline on a realistic path planning application, we collected a dataset in an underground mine consisting of a room network hewn from the rock. The dataset was collected with the same sensor platform as in NCD, but mounted on a Husky wheeled robot. We ran the pipeline with 6.5 cm resolution and 60 m range again. The resultant occupancy map was then used by an RRT* [30] path planner to compute the shortest collision free path between two locations.

The result is presented in Fig. 8: the volumetric reconstruction is highly detailed, giving clear definition even in narrow doorways and corridors. This allowed the path planner to find the optimal path to the goal despite obstacles such as support pillars.

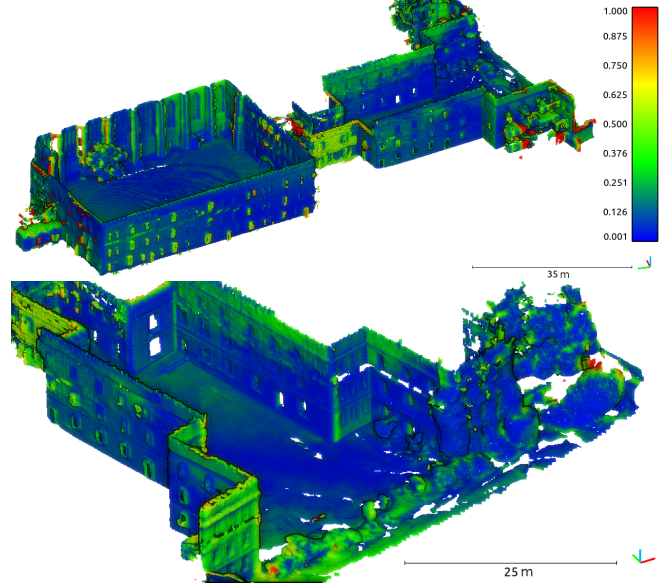


Fig. 7: Evaluation of reconstruction accuracy using point-to-point distance compared against ground truth.

VI. CONCLUSION AND FUTURE WORK

To summarise, our proposed system can efficiently reconstruct large-scale environments at 3 Hz with high resolution (~ 5 cm) using long range LiDAR scans (60 m max range). The core data structure of *supereight* exploits sparsity of the environment with an adaptive resolution representation and allows for better scalability and efficiency in scan integration as compared to state-of-the-art mapping systems such as Voxgraph and OctoMap.

We use submaps to introduce elasticity into our volumetric map. This allows our reconstruction to be corrected with SLAM loop closures during long exploration tasks. Our results demonstrate global consistency in the map when compared to the ground truth. Graph sparsification and

¹<https://www.danielgm.net/cc/>

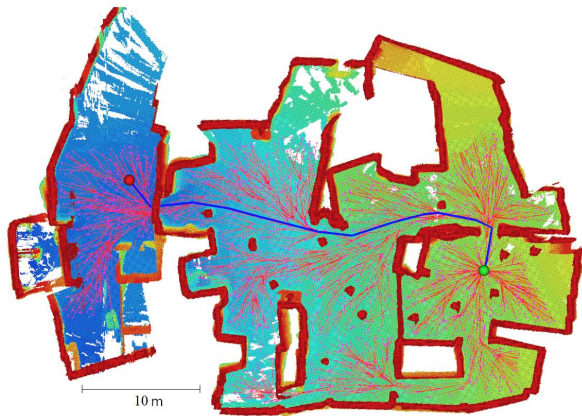


Fig. 8: Using our reconstruction result for path planning in an underground room network. Green sphere - start; red sphere - end; red tree with magenta nodes - RRT*; blue trajectory - planned path.

submap fusion also allow our reconstruction to scale with the size of the environment instead of the duration of exploration.

Our reconstruction result is suitable for robotic operations such as path planning even in typically challenging situations such as narrow doorways and clutter thanks to the high resolution reconstruction around obstacles.

To further improve the reconstruction accuracy, we plan to extend *supereight* to incorporate a undistorted spherical model to account for LiDAR motion distortion and enable high speed operation. In addition, we aim to improve the submap clustering so that the number of maps plateaus when a space is fully explored.

REFERENCES

- [1] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon path planning for 3D exploration and surface inspection," *Autonomous Robots*, vol. 42, no. 2, pp. 291–306, 2018.
- [2] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an MAV," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Paris, France, June 2020.
- [3] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.
- [4] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme, "Active planning for underwater inspection and the benefit of adaptivity," *Intl. J. of Robotics Research*, vol. 32, no. 1, pp. 3–18, 2013.
- [5] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *Intl. J. of Robotics Research*, vol. 31, pp. 1445–1464, 2016.
- [6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems (RSS)*, vol. 2, no. 9, 2014.
- [7] E. Vespa, N. Funk, P. H. J. Kelly, and S. Leutenegger, "Adaptive-resolution octree-based volumetric SLAM," in *Intl. Conf. on 3D Vision*, 2019, pp. 654–662.
- [8] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An Atlas framework for scalable mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2003, pp. 1899–1906 vol.2.
- [9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE Intl. Symp. on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.
- [10] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems (RSS)*.
- [11] C. Park, P. Moghadam, J. Williams, S. Kim, S. Sridharan, and C. Foakes, "Elasticity meets continuous-time: Map-centric dense 3D LiDAR SLAM," *arXiv preprint arXiv:2008.02274*, 2020.
- [12] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 2021–2028.
- [13] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "Octree-based fusion for realtime 3D reconstruction," *Graphical Models*, vol. 75, no. 3, pp. 126 – 136, 2013.
- [14] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields," in *Robotics: Science and Systems (RSS)*, vol. 4, 2015, p. 1.
- [15] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 1, 2017.
- [16] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-Time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.
- [17] M. Tanner, P. Piniés, L. M. Paz, Ștefan Săftescu, A. Bewley, E. Jonasson, and P. Newman, "Large-scale outdoor scene reconstruction and correction with vision," *Intl. J. of Robotics Research*, vol. 0, no. 0, p. 0278364920937052, 2018.
- [18] D. De Gregorio and L. Di Stefano, "Skimap: An efficient mapping framework for robot navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 2569–2576.
- [19] B.-j. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3D environments," *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, no. 2, pp. 2175–2182, 2018.
- [20] P. Sodhi, B.-J. Ho, and M. Kaess, "Online and consistent occupancy grid mapping for planning in unknown environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 11 2019, pp. 7879–7886.
- [21] D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," 2020.
- [22] J. Nieto, J. Guivant, and E. Nebot, "Denseslam: Simultaneous localization and dense mapping," *Intl. J. of Robotics Research*, vol. 25, no. 8, pp. 711–744, 2006.
- [23] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, 2020.
- [24] 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, no. 3, pp. 189–206, Apr. 2013.
- [25] M. Ramezani, G. Tinchev, E. Iuganov, and M. Fallon, "Online LiDAR-SLAM for legged robots with robust registration and deep-learned loop closure," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 4158–4164.
- [26] N. Funk, T. Juan, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger, "Multi-resolution 3d mapping with explicit free space representation for fast and accurate mobile robot motion planning," *arXiv preprint*, 2020.
- [27] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1144–1151, Apr. 2018.
- [28] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou, "A closed-form Bayesian fusion equation using occupancy probabilities," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 380–388.
- [29] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The Newer College Dataset: Handheld LiDAR, inertial and vision with ground truth," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.
- [30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Intl. J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.