GEM: online globally consistent dense elevation mapping for unstructured terrain

Yiyuan Pan, Xuecheng Xu, Xiaqing Ding, Shoudong Huang, Yue Wang, and Rong Xiong

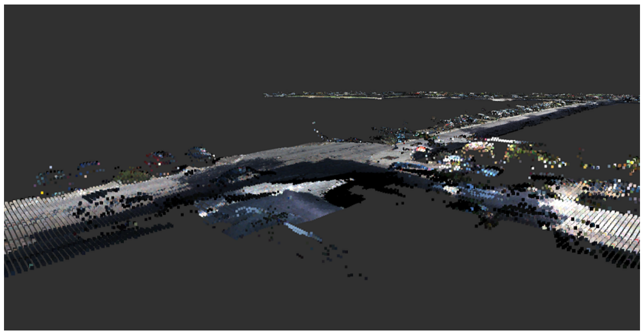
***Abstract*—Online dense mapping gives a representation of the unstructured terrain, which is indispensable for safe robotic motion planning. In this paper, we propose such an elevation mapping system, namely GEM, to generate a dense local elevation map in constant real-time for fast responsive local planning, and maintain a globally consistent dense map for path routing at the same time. We model the global elevation map as a collection of submaps. When the trajectory estimation of the robot is corrected by SLAM, only relative poses between submaps are updated without re-building the submap. As a result, this deformable global dense map representation is able to keep the global consistency online. Besides, we accelerate the local mapping by integrating traversability analysis into the mapping system to save the computation cost by obstacle awareness. The system is implemented by CPU-GPU coordinated processing to guarantee constant real-time performance for in-time handling of dynamic obstacles. Substantial experimental results on both simulated and real-world dataset validate the efficiency and effectiveness of GEM.**

***Index Terms*—elevation mapping, consistency, scalability**

I. Introduction

PERCEIVING the terrain and nearby obstacles is a vital capacity for safe and efficient on-the-ground autonomous navigation . This task is achieved by building a map of the environment based on sensor measurements, which is an indispensable component in mobile robot navigation system. Simultaneous localization and mapping (SLAM) is such a solution to integrate the sensor data into a map representation in real time [1]. Thanks to the full trajectory optimization in SLAM, the yielded map is globally consistent. However, the map in SLAM is generally encoded by a set of features [2] or point cloud [3], which is sparse and unsuitable for motion planning.

To address the problem, several mapping methods are proposed to build a dense map based on the trajectory estimation of SLAM using acquired sensor data. Occupancy grid map is the most mature representation, in which each grid indicates whether the corresponding space is free or occupied [4]. Many mature planning methods are developed on this type of map [5]. However, occupancy grid map cannot deal with the uneven ground surface, like slope, thus mostly used in the indoor environment. Elevation map [6] replaces the occupancy information stored in each grid with the elevation, which is able to represent the 2.5D ground surface by grid map. As a result, elevation map is widely applied in on-the-ground navigation, especially on unstructured terrain [7].


(a) Dense mapping before loop correction using LiDAR.

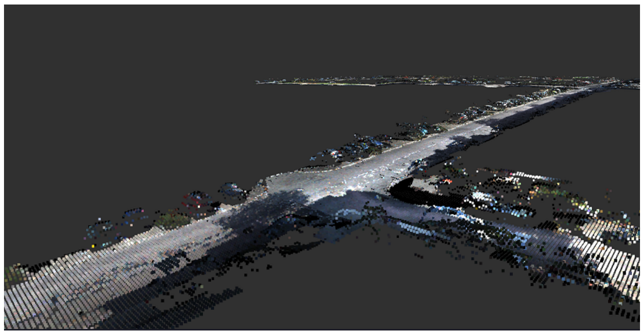

(b) Dense mapping after loop correction using LiDAR.

Fig. 1. Our system, GEM, can build large-scale globally consistent dense map online. The results show the performance of mapping KITTI odometry 00 using LiDAR before and after loop closure correction. Note that the edge of shadow is very smooth in the updated map.

Generally, a global elevation map is built offline, when the whole trajectory in the mapping stage is estimated. However, in scenarios where no prior map is available, the whole trajectory is estimated and corrected by SLAM online, which means we have to save all sensor data and re-build the elevation map using the current trajectory. Note that such corrections occur frequently during SLAM, preventing the system from mapping the environment online to capture the dynamic obstacles in time. In [8], robocentric elevation mapping is proposed to build a map using only visual odometry (VO) or visual-inertial system (VINS), which does not correct the trajectory, avoiding the map re-building to achieve real-time performance. But such a solution loses the global consistency of the dense map, which may confuse the global path routing. Therefore, the main challenge for mapping is to achieve global consistency and real-time performance at the same time.

In this paper, we introduce an online globally consistent

This work was supported in part by the National Key R&D Program of China (2019YFB1309503), and in part by the National Nature Science Foundation of China (61903332).

Yiyuan Pan, Xuecheng Xu, Xiaqing Ding, Yue Wang, and Rong Xiong are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou, P.R. China. Shoudong Huang is with the Center for Autonomous Systems (CAS), the University of Technology, Sydney, Australia. Yue Wang is the corresponding author `wangyue@iipc.zju.edu.cn`.

elevation mapping system using either stereo vision or LiDAR, namely GEM. The central idea is to represent the global map by a set of submaps. For global consistency, the relative poses between submaps are updated with each submap keeping invariant, deforming the map online according to the corrected trajectory without re-building the map as shown in Fig. 1, thus making online processing possible. For real-time performance, as only surrounding environment is important for local planning, we propose a local mapping stage decoupled from submap and global map building stage, which is guaranteed in exactly constant size. Furthermore, we integrate the traversable region detection into local mapping to reduce the computation cost by only checking obstacles in ray tracing. The whole system is implemented by GPU-CPU coordinated processing, simultaneously achieving in-time response to dynamic obstacles, and globally consistent mapping for long-horizon path routing. In addition, we also present a mechanism to relate the number of submaps to the size of the mapping area, enabling long-term scalability. To the best of our knowledge, this is the first elevation mapping system achieving both goals. The contributions are summarized as:

- Formulating a global elevation map as a set of submaps, so that we can deform the global map to keep consistency without re-building the map upon a trajectory correction;
- Presenting a framework having decoupled local mapping, submapping and global mapping to exactly bound the computation complexity at each step, as well as obstacle-aware ray tracing and map maintenance mechanism to further improve efficiency and scalability;
- Building a GPU-CPU coordinated mapping system, namely GEM, which has constant real-time local mapping and globally consistent mapping capability. Both stereo vision and LiDAR are applicable to the system. The code of GEM is released[1];
- Conducting experiments on both simulation and real world involving on-the-road and campus environments with dynamic objects are presented to validate the effectiveness of GEM. A video of the mapping process is uploaded to [9].

II. Related Work

A. SLAM and sparse mapping

There has been extensive work on SLAM, which generally uses a sparse map to model the environment. When visual sensors are employed, the feature points are integrated with the trajectory into the SLAM estimation, resulting in a globally consistent sparse feature map. A typical method following this idea is ORB-SLAM2 [2]. As globally consistent estimation takes much time, only trajectory is considered in SLAM estimation, like VINSmono [10]. To maintain the global consistency, place recognition[11], [12] is used to provide loop closure candidates and both classes of SLAM methods correct the whole trajectory when a loop closure happens. To avoid frequent correction of the trajectory, there are methods only estimating the current poses [13]. As a result, the sparse map keeps invariant all the time. But such solutions are actually visual odometry, losing the global consistency in the map. For LiDAR, there are also similar situations. For globally consistent SLAM methods [14], the trajectory changes during mapping session, while LiDAR inertial odometry, keeps invariant with a cost of losing global consistency [15]. In addition, it is hard using either LiDAR or vision sensors for motion planning on the sparse maps with both kinds of SLAM algorithms.

[1]https://github.com/ZJU-Robotics-Lab/GEM

B. Dense mapping and representations

To build a dense map which is required by many motion planning methods, several types of dense mapping methods are proposed. The most mature method is the occupancy grid map, in which each grid indicates the probability of free [4]. The estimation of the grid state is achieved by sequentially integrating the sensor data following the Bayesian filtering. However, the occupancy grid map is only suitable for planar ground. To extend dense mapping for the environment with various elevations, 2.5D and 3D dense mapping methods are proposed. Elevation mapping is an intuitive extension of the occupancy grid map [6]. It still uses a 2D grid map, but estimates the ground elevation in each grid to represent the 2.5D uneven terrain. Though, elevation mapping cannot model the environment with multiple layers, it is sufficient for on-the-ground navigation by only considering the obstacles lower than the robot height. As for constructing elevation map with uncertainty modeling, early works on the generation of elevation map are presented by [16], [17], they apply special filtering algorithms to eliminate measurement errors yet ignoring the error propagation from pose estimation. In the method presented by [18], the uncertainty of the robot's estimated pose is reflected in the map by linearly growing the variance of the grid height. [8] constructs a gravity-aligned elevation map and further takes the effect of in-plane uncertainty of the map from the uncertainty of estimated poses into account. For 3D navigation, like underwater or aerial robots, building a 3D dense map is necessary. There are methods to estimate occupancy by extending the occupancy grid map to 3D occupancy volume [19]. To achieve sub-voxel accuracy [20], [21], truncated signed distance function (TSDF) is used. They use a 3D volume to represent the environment, in which each voxel stores the signed distance to the surface, enabling multi-frame integration on the distance. In addition, surfel based map is also an option, in which each surfel is a small disk to achieve higher accuracy [22]. Note that the most common type of dense map for motion planning is grid map, like the 2D and 3D occupancy map, as well as the elevation map [23]. The other 3D representations generally call for additional conversion to the occupancy grid or other intermediate representation for planning [24]. More importantly, these dense mapping methods are applied after the globally consistent trajectory is no longer corrected by SLAM, hence offline, or applied when the trajectory is yielded by odometry, thus lose global consistency.

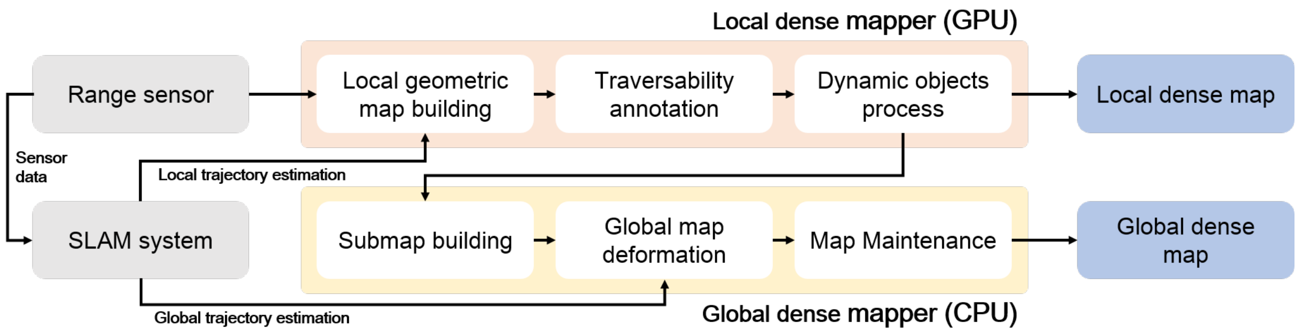


Fig. 2. The architecture of the proposed GEM. Any sensor with range measurements can be applied as input, e.g. LiDAR. The SLAM system provides local and globally consistent trajectory estimations. The complete system consists of two threads, the local dense mapper and the global dense mapper. The local dense mapper yields the real-time robocentric elevation map for motion planning, while the global dense mapper, a globally consistent dense map for global routing.

C. Dense mapping with online SLAM corrections

Several methods have proposed to maintain the global consistency of the map when online SLAM correction happens. The main idea is to model a deformable dense map representation. The first line of works connects the elements of the dense map with the trajectory, thus formulating a similar problem like feature based SLAM. Surfel-based method [25] is proposed to register consecutive 3D laser scans with dense map to estimate the robot motion and construct a pose graph for global optimization, which is similar with [6] based on the elevation dense map. In [26], [**?**], the dense map is represented as a graph with each node being a surfel. The map graph is then combined with the trajectory for optimization to achieve globally consistent result. These methods have the most consistent results, but the computation cost constrains the scalability to a larger environment [27]. The second line of works follows the idea of trajectory only estimation. The main idea is to partition the dense map into submaps, which are regarded as local observations for each pose in the trajectory, making the map deformable. Following this idea, several works extend the one TSDF volume to multiple TSDF subvolumes as a collection of submaps, thus useful for the large-scale environment [28]. The similar idea is also utilized in occupancy volumes for globally consistent dense map [29]. These methods build a new submap by only considering the temporal closeness, leading to the linearly growing complexity of the map, constraining the scalability even in a small area when a robot moves in a loop for long term. To solve this problem, there are works that consider the spatial closeness. For TSDF, C-blox [27] is proposed by integrating the measurement to previously built submap when a loop closure happens. This strategy is also integrated into surfel based mapping [30].

Compared with these methods, the proposed mapping system follows the idea of submap based mapping to build an elevation map. Except for the redundant information and extra conversion when utilizing 3D dense map for on-the-ground navigation, we consider that mapping for navigation has several other aspects. First, accurate reconstruction of the obstacle is unnecessary for planning, but obstacle awareness is not easily integrated into the TSDF mapping [31]. Then surfel based mapping only considers the end points of the sensor data, losing dynamic objects in the map, which are important for safe navigation. Also, the dense map can be rendered as orthomosaic image for collaborative localization [32], [33]. As a summary, we expect the proposed system to not only bridge the gap between on-the-ground navigation requirements and dense mapping systems but also boost the development of multi-agent SLAM.

III. System Overview

The architecture of the proposed GEM is shown in Fig. 2. The input consists of the SLAM local and global trajectory estimation, as well as the synchronized sensor data. Here local trajectory is estimated by sensor-based odometry, like visual inertial odometry, while global trajectory, by the global optimization in the SLAM system. In addition, GEM is aware of the trigger signal from the SLAM system when a global optimization is executed, so that the global consistency is maintained in time. An arbitrary SLAM system with such output is compatible with GEM. We show results from GEM with a LiDAR and a stereo visual SLAM as the input in the experimental results. Provided by this information, GEM yields a robocentric elevation map for local motion planning and a global elevation map for routing. The system has two threads, the local dense mapper builds the dense robocentric elevation map, while the global dense mapper deforms the graph to maintain the global consistency of the map.

Local dense mapper. In the local dense mapper, there are three steps to generate robocentric elevation map, including dense mapping construction, traversable region detection, and dynamic objects processing. The robocentric elevation map builds a geometric elevation map with the frame moving with the robot. The label of the geometric is annotated by the traversable region detection. If there are dynamic objects, dynamic objects process updates them in the map to track the current situation. In order to ensure real-time performance, we propose a selectively updating mechanism and design a GPU implementation.

Global dense mapper. In the global dense mapper, there are three components: submap building, map deformation and submap maintenance. The global dense mapper integrates the local map to existing submap or creates a new submap on the fly. Map deformation is triggered by the SLAM system. It correspondingly transforms the frames of submaps using the corrected trajectory from SLAM to eliminate drift. Then if a completed submap has a large overlap with the existing submaps, say a loop closure happens, the submap maintenance integrates the new submap to the existing submaps to avoid ever-growing complexity with respect to the trajectory length. Instead, the computation complexity of this thread depends on the environment size, increasing the scalability.

IV. Local dense mapper

Algorithm 1 Local dense mapping

```
Input:
    {r_t}: sensor data, T_t: the pose, M_t: the map frame
Output:
    L_t: the local elevation map
1:  // Uncertainty propagation
2:  in parallel do
3:      $i = threadIdx.x + \alpha \times blockIdx.x$[1]
4:      $e_i \leftarrow Transform(r_{t,i}, T_t)$
5:      $\sigma_e^2 \leftarrow Calculate\_variance(r_{t,i}, T_t)$
6:  end
7:  // Fuse multiple observations
8:  in parallel do
9:      $i = threadIdx.x + \alpha \times blockIdx.x$
10:     for all $e_i$ do
11:         if $e_i$ lies in $E_i$ then
12:             $L_i \leftarrow Fuse\_observations(M_t, L_i, e_i, \sigma_e^2)$
13:         end if
14:     end for
15: end
16: // Annotate traversability
17: in parallel do
18:     $i = threadIdx.x + \alpha \times blockIdx.x$
19:     $v_{i,slope} \leftarrow Calculate\_slope(E_i)$
20:     $v_{i,rough} \leftarrow Calculate\_roughness(E_i)$
21:     $L_{i,free} \leftarrow Calculate\_trav(v_{i,slope}, v_{i,rough})$
22: end
23: // Process dynamic objects
24: in parallel do
25:     $i = threadIdx.x + \alpha \times blockIdx.x$
26:     if $Trav_{E_i}$ < Threshold then
27:         $Flag = Ray\_tracing(E_i, \{r_t\})$
28:         if $Flag = Dynamic$ then
29:             $Initialize(E_i)$
30:         end if
31:     end if
32: end
```

We first introduce the structure of the elevation map. The elevation map is defined on a grid map with each grid indexing the elevation and other information. The grid index is discrete. The frame system is shown in Fig. 3. The pose of the sensor is denoted as T_t with respect to the global frame G. The transformation of the map frame with respect to G is defined as M_t, which has the same rotation and height as G, and the same horizontal translation as T_t. Therefore, M_t is always lying in the $xy-$plane of G. T_t is updated by the local trajectory estimation from SLAM. Since local trajectory estimation is very smooth, not corrected by SLAM, we can safely use it to aggregate the elevation map L_t. In this framework, all sensors with range measurements denoted as points $\{r_t\}$, can be fused in T_t to build the dense map. The algorithm of local dense mapping is shown in Alg.1.

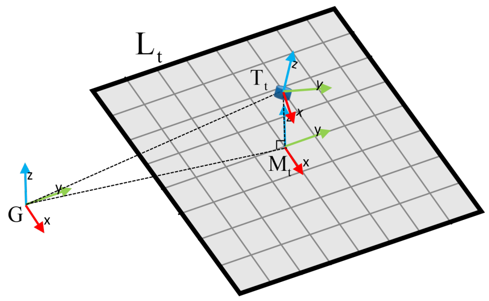


Fig. 3. Frame system for the local dense mapper. The global map is in global frame G, the sensor frame is in T_t. The local map frame M_t has the same rotation and height as G with fixed relative xy translation to T_t.

A. *Local geometric map building*

Define the local map L_t on grid map E spanned at M, where E_i is a grid in E. In each grid, we store color $L_{i,color} \in \mathcal{R}^3$, elevation $L_{i,e} \in \mathcal{R}$, variance $L_{i,var} \in \mathcal{R}$ and $L_{i,free} \in \mathcal{R}$. To build the elevation map, we first transform each point measurement r_t to G by $T_t r_t$. As the local map frame is moving with the robot but aligned with G, we can simply select the first two entries of points as the index of E, on which we can render the elevation map and check the points out of the region of L_t at each timestep. Though T_t has drift in the global frame, points in robocentric region are locally consistent. We parallelize the steps in local mapping to respond to real-time environment dynamics.

Measurement model of elevation map. We first introduce the uncertainty modeling of the elevation. Given a measurement r_t in the sensor frame, we transform it into the global frame point.

$$e = PT_t r_t \tag{1}$$

where $P = [0, 0, 1]$ extracts the 3rd entry of the global point as the elevation e, while the first two entries are regarded as the continuous index. To obtain the variance of the elevation for multiple measurements fusion, the Jacobian matrices of the measurement J_s, the sensor rotation J_Φ and translation J_t are calculated as follows.

$$J_r = \frac{\partial e}{\partial r_t} = PR_t \tag{2}$$

$$J_\phi = \frac{\partial e}{\partial \phi} = -PR_t(r_t)^\times \tag{3}$$

where $\cdot^\times$ represents the generated skew-symmetric matrix of $\cdot$, R_t denotes the rotation part of T_t, ϕ is the small perturbation on the rotation $R_t = \bar{R}_t(I + \phi^\times)$ with $\bar{R}_t$ the current estimated rotation matrix. With linear propagation, we have the uncertainty of the elevation as

$$\sigma_e^2 = J_r \Sigma_r J_r^T + J_\phi \Sigma_\phi J_\phi^T \tag{4}$$

[1]α: thread number per block x-dimension, $threadIdx$: CUDA thread index, $blockIdx$: CUDA block index

where Σ_r, obtained from range sensor noise models, represents the covariance matrix of the measured point, Σ_ϕ denotes the covariance matrix of the sensor rotation acquired from the uncertainties of the pose estimator, like visual SLAM. In this way, each single measurement gets its continuous grid index, elevation and uncertainty. Since only planar translation exists between G and M, the elevation uncertainty keeps the same. The parallelization of this step is shown in line 2-6 of Alg.1.

Multiple elevation measurements fusion. When rendering the robocentric elevation map L_t, we fuse multiple e lying in the same grid E_i. Specifically, we first apply the χ^2 test to evaluate the difference Δ between the highest elevation e measured at the most recent timestep and the corresponding grid elevation e_g with variance σ_g:

$$\Delta = \frac{|e_g - e|}{\sigma_g} \tag{5}$$

If Δ is not passed and higher elevation is measured, the grid elevation is re-initialized. Then we fuse the points passing the test weighted by the inverse of variance, namely variance weighted fusion in the sequel, to obtain an updated estimation (e_g, σ_g^2) as follows:

$$e_g = \frac{\sigma_e^2 e_g + \sigma_g^2 e}{\sigma_g^2 + \sigma_e^2} \qquad \sigma_g^2 = \frac{\sigma_g^2 \sigma_e^2}{\sigma_g^2 + \sigma_e^2} \tag{6}$$

The reason to apply χ^2 test is to consider dynamic objects and vertical obstacles. Finally, points lying in the region of L_t are used to generate L_{color}, L_e and L_{var}. As shown in line 8-15 of Alg.1, we parallelize this fusion step with respect to each grid. Note that the derivation of uncertainty is similar to [8] in local dense mapping, but we do not assume IMU aided SLAM system, which may cause the drift of pitch and roll angle of estimated poses in long term. To eliminate the drift, we apply global optimization and introduce submap frame, which is presented in Section V.

B. Traversability annotation

After generating the geometry of the dense map, the free region L_{free} on the elevation map is annotated. We use normal vector and height deviation as the features of each grid E_i. Normal vector reflects the slope of the terrain, which is computed by fitting a local planar patch [34]. Height deviation is calculated by differentiating the elevation in the neighborhood to reflect the roughness of the terrain. Following [7], we estimate the free traversability for each grid by

$$v = max(0, 1 - w_s \frac{v_{slope}}{v_{s_{crit}}} - w_r \frac{v_{rough}}{v_{r_{crit}}}) \tag{7}$$

where v_{slope} and v_{rough} denotes the normal and height deviation based slope and roughness, w_s and w_r denotes the weight factor of slope and roughness, $v_{s_{crit}}$ and $v_{r_{crit}}$ represents the threshold of the corresponding feature when the grid is traversable. If the score is higher than a threshold, we annotate the grid as traversable to form L_{free}. We parallelize the annotation step with respect to each grid in line 17-22 of Alg.1. Note that the parameters except w_{slope}, w_{rough} are determined by the types of robots, which are given in Section VI.

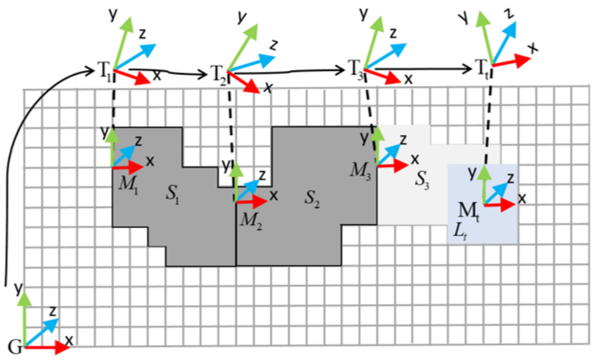


Fig. 4. Frame system for the global dense mapper. S_1 and S_2 are compeleted submaps, S_3 is an active submap. Points leaving L_t are stored into the submap S_3 defined in frame T_3 at that moment.

C. Dynamic objects process

The dense mapping method above utilizes only end point model, which can capture the obstacles, but cannot clear the obstacles, thus dynamic objects may clutter the local elevation map. To address the problem, we apply ray tracing, whose principle is that no obstacles lying in front of the end point of a ray. Following this idea, we generate a ray for each r_t to clear the potential dynamic objects in L_t built above. For a grid crossed by a ray, if the elevation is higher than the ray, the obstacle in that grid can be regarded as a passed dynamic object and all data of that grid is initialized. As we have annotations L_{free} for each grid, we only evaluate the obstacle regions crossed by rays, which greatly reduces the computation cost but still performs compelling results. In line 24-32, as the ray casting for each grid is independent, we parallelize the process with respect to each grid.

V. Global dense mapper

As shown in Fig. 4, the local robocentric elevation map L_t is sent to the global dense mapper. If the length of the trajectory building the current submap exceeds the threshold, 20m in our experiment, a new submap S_k is made and initialized by L_t with submap frame $T_k = T_t$, otherwise, we integrate L_t to an existing submap, say S_k. Note that the frames of all submaps $\{S_k\}$ are determined by $\{T_k\}$, and the main idea of the global dense mapper is to deform the global map by transforming the frames of all submaps according to the trajectory corrected by the SLAM system online.

A. Submap building

We define a submap as $S = [S_{color}, S_{point}, S_{var}]$ where $S_{color} \in \mathcal{R}^{3\times N}$ denotes the color of the points, $S_{point} \in \mathcal{R}^{3\times N}$ denotes the points in the submap, $S_{var} \in \mathcal{R}^{3\times 3\times N}$ denotes the covariance matrix of the points, and N is the number of points stored in the submap. The submap points are aggregated from points that leave the region of L_t, denoted as p_t. Specifically, given a p_t, we transform it to submap frame T_k by $s_{point,k} = T_k^{-1} M_t p_t$ and denote $s = M_t p_t$ for clear derivation. The Jacobian matrices of the measurement J_s, the sensor rotation J_Φ and translation J_t are

$$J_s = \frac{\partial s_{point,k}}{\partial s} = R_k^T \tag{8}$$

$$J_\phi = \frac{\partial s_{point,k}}{\partial \phi} = [R_k^T(s - t_k)]^\times \tag{9}$$

$$J_t = \frac{\partial s_{point,k}}{\partial t_k} = -R_k^T \tag{10}$$

where R_k and t_k are rotation and translation part of T_k, ϕ is the small perturbation on the rotation as before. The uncertainty propagation is given as

$$s_{var} = J_s \Sigma_s J_s^T + J_\phi \Sigma_\phi J_\phi^T + J_t \Sigma_t J_t^T \tag{11}$$

Since these points are updated in several timesteps, we only keep points indicating traversable terrain in the submap with confidence. This selection is motivated by the fact that on-the-ground motion planning algorithms do not consider the accurate shape of untraversable obstacles, thus saving the storage and computation cost.

Hash grids for unorganized points. When we save unorganized points, say $s_{point,k}$, the dense 2D grid structure loses. If we still use a 2D array to save these points, as the size of the submap cannot be known before the completion of submap building, we cannot determine the memory to be allocated for the 2D array. Instead, we use a hash table for unorganized submap points storage, achieving $O(1)$ access complexity and keeping the grid index. For each point $s_{point,k}$, we create a new grid in the hash table if no stored points share the same grid index, otherwise, we update the stored point using variance weighted average. Finally, when the submap building is completed, we vectorize the points in hash grids into S with their colors and variances to keep the storage compact.

B. Global map deformation

When a SLAM correction happens after the detection of a loop closure, the thread of global map optimization is triggered by a new globally consistent trajectory estimation. As the frames of submaps $\{S_k\}$ are determined by $\{T_k\}$, which is a subset of the updated trajectory, we can deform the global map using the most recent global trajectory estimation. Specifically, for each submap S_k, we simply transform the points by $T_k S_{point,k}$, but keep $S_{color,k}$ and $S_{var,k}$ invariant. As a whole, the collection of transformed submaps $\{T_k S_{point,k}\}$ are represented in the unified global frame G, building a global map.

Render a global dense map. For running global motion planning algorithms, we have to generate a global dense elevation map, which is achieved by projecting $\{T_k S_{point,k}\}$ to the grid map defined by the global frame G. Different from [8], we do not assume a global observable pitch and roll estimation, failing to decouple the uncertainty, but applying 6DoF correction for global consistent mapping instead of 4DoF. Therefore, we have to deal with changed roll and pitch of submap frames T_k, which may cause a changed mapping between points and grid index. When multiple points are projected into one grid, we apply the variance weighted average to fuse the elevation at that grid. Finally, a global dense elevation map is rendered.

C. Map Maintenance

With the method above, the complexity of the global map is unbounded, since the number of submaps grows with respect to the trajectory length. Even the trajectory is looping in a limited region, this problem still exists. We propose a strategy to maintain the complexity of the map by relating the number of submaps to the region size. Given a newly completed submap, say S_k, we find its nearest submap, say S_i. If the two submaps have larger overlap than a threshold, we fuse S_k to S_i to reduce the complexity.

Given T_i and T_k, the frames of S_i and S_k, we have

$$S_{point,i,k} = T_i^{-1} T_k S_{point,k} \tag{12}$$

Considering that inactive submap is stored as vectors, we re-activate the submap by creating hash grids again to assign the points with grid indices. For fusion, we follow the same mechanism as that in submap building, by regarding $S_{point,i,k}$ as a set of measurement points for S_i. If there is a large time gap between S_i and S_k, we can simply assign the $S_{point,i,k}$ to corresponding grids without weighted average as the environment may change. For non-overlapping part, new hash grids in S_i are created to hold $S_{point,i,k}$. As non-overlapping part is constrained by the threshold, S_i does not expand unboundedly.

D. Threads management

A possible scenario is that a robot revisits the past area many times in a small time interval, causing frequent loop closures to request map deformation multiple times in a short duration. As a result, the global mapper thread may congest the local mapper, causing an unacceptable delay of local mapping for safety issues. To solve this problem, we follow the local-global threads management mechanism in state-of-the-art SLAM systems, like ORB-SLAM2 [2]. When a new request to global mapper occurs, it first stops the current running map deformation if the last request is being processed, then restarts the map deformation using the new global trajectory yielded by SLAM. Therefore, the new global dense map is generally built after the last loop closure in this time interval where frequent loop closures occur. Such a mechanism can keep the local mapping thread running in real-time, which is similar to the local trajectory estimator in the real-time SLAM system. At the perspective of application, since global consistent map is crucial for global path planning, which cannot be re-planned frequently, we consider that such stop-and-restart mechanism does not impact the robot operating in the real world.

VI. Implementation details

The local dense mapper is implemented on GPU, and the global dense mapper is implemented on CPU to parallelize the computation for real-time performance. The traversability annotations depend on the physical properties of the robot. In this paper, the traversability is judged according to the vehicles, and we set $w_s = 0.5$, $w_r = 0.5$, $w_{s_{crit}} = 0.6$, $w_{r_{crit}} = 0.2$ in Eq. 7 and the threshold as 0.7 of L_{free} empirically. Of course, the traversability can also be tuned for other types of robots e.g. legged robots in [7]. We set a

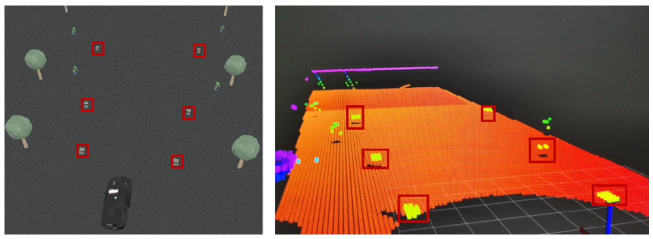


Fig. 5. The simulation environment (left) and the corresponding mapping result using GEM (right).

distance-based criterion for submap creation by comparing the length of the trajectory segment in the submap with $20m$. The whole system is implemented on a platform with Intel i7-8700 (CPU) and NVIDIA 1060 (GPU). We also implement the local dense mapper on NVIDIA TX2 for on-board processing. It is an embedded system which is very useful for mobile robots with limited power supply. Our system is written in C++ and integrated into the Robot Operating System (ROS).

VII. Experiments

In the experiments, we evaluate GEM from four aspects: feasibility for the on-the-ground navigation, time analysis of the local and global mapping, global consistency and scalability. For comparative study, four methods with publicly released codes are utilized: (i) Octomap, an octree based 3D dense mapping system with local consistency [35], (ii) C-blox, a TSDF volume based 3D dense mapping system which releases local consistency implementation [27], (iii) Surfel-Mapping, an unorganized surfel based 3D mapping system with global consistency [30] and (iv) ElasticFusion, a surfel based 3D dense mapping system with CPU-GPU coordination. Note that the original pose tracker of ElasticFusion cannot work in a large-scale environment, so we adjust its code to be compatible with the interface of ORB-SLAM2. Thus all vision mapping systems are kept with the same input trajectory estimator. Three datasets are utilized for evaluation. A simulation dataset with known structure is used for accuracy testing.

Public real-world dataset KITTI [36], which is collected in an on-the-road environment, is employed for testing the global consistency, real-time performance and scalability. Both stereo vision and Velodyne-64 LiDAR are available in this dataset. The second real-world dataset collected by ourselves in the campus environment, namely YQ, is also utilized for testing the global consistency and real-time performance. On the YQ dataset, the data is acquired by two VLP-16 LiDARs installed on a mobile robot. One LiDAR is utilized for pose estimation, while the other for mapping. In addition, a small simulation is built in V-rep [37], which aims at evaluating the accuracy as the ground truth of this environment is available. For the SLAM system, ORB-SLAM2 is used for vision sensors, and LiDAR SLAM as that in [38] is used for LiDAR sensors. We modify the ORB-SLAM2 output to satisfy our interface requirement.

A. Local map quality

The primary indicator for mapping is accuracy since ground surface is important for traversability, which is highly related to the safety of navigation. To evaluate the accuracy, we put six cubes with the size of $0.5 \times 0.5 \times 0.5m^3$ in the simulation environment, so that we can determine the marker points for distance measurement easily. LiDAR is employed for data acquirement. The simulation environment and the mapping result are shown in Fig. 5. Four resolutions of $0.1m$, $0.2m$, $0.3m$ and $0.4m$ are set to evaluate the accuracy with respect to the resolution. All methods are fed with the ground truth trajectory and 64-ring 3D LiDAR scans, keeping the same input for all experiments. Octomap, C-blox and ElasticFusion (a surfel based method with no resolution setting) are employed as a comparison.

Height accuracy. We manually pick the cubes from the built map, and calculate the height of the cubes above the ground surface. By comparing the height with the ground truth i.e. $0.5m$, we have the elevation accuracy. The results are shown in Tab. I, the error of object height is around $5mm$ and not affected by the resolution using GEM. This error has almost no impact on traversability analysis. As a comparison, the elevation accuracy is affected by the resolution using Octomap and C-blox due to the 3D discretization and ElasticFusion achieves competitive elevation accuracy with GEM owning to the continuous representation.

Obstacle accuracy. The obstacle to robot distance is also important for obstacle avoidance. We manually pick a cube as a reference and measure the distances from other cubes to this reference cube in both simulation and built maps. The results are shown in Tab. I. Using the continuous 3D surfel based representation, ElasticFusion achieves the best obstacle accuracy. However, as ElasticFusion highly depends on the sensory data density, it fails to map all 6 obstacles (only 4 detected) for lack of sufficient competitive observation with the sparse range sensor, like LiDAR. Note that other methods give similar sub-resolution accuracies. As a planning algorithm runs in resolution level, this error level is sufficient. As GEM has only 2-dimensional grid but with higher height accuracy which is useful for complex terrain analysis, we verify that GEM is more suitable for on-the-ground navigation.

TABLE I
LOCAL DENSE MAP ACCURACY

Resolution(cm)	10	20	30	40	Avg	#Det.[a]
GEM ele-error	0.46	0.40	0.51	0.56	**0.48**	**6**
GEM obs-error	10.38	13.57	14.19	14.21	13.09	
Octomap ele-error	0.00	10.00	11.67	13.33	8.75	**6**
Octomap obs-error	11.34	12.61	14.75	15.51	13.55	
C-blox ele-error	2.73	8.72	15.34	18.68	11.37	**6**
C-blox obs-error	9.51	11.98	18.67	25.38	16.39	
EF[b] ele-error			0.79			4
EF obs-error			**8.05**			

[a] #Det.: The number of detected cubes. [b] EF: ElasticFusion.

TABLE II
LOCAL DENSE MAPPER STEP-BY-STEP TIME ANALYSIS

Steps	EM[8]	GEM	GEM
Hardware	PC	PC	TX2
Geometric map building (ms)	256	**20**	66
Traversability annotation (ms)	-	**15**	49
Dynamic objects process (ms)	893	**1**	4
Total time (ms)	548	**48**	171

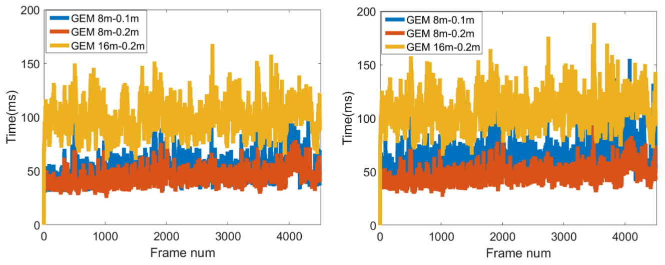


Fig. 6. Time evolution trend of GEM for the local dense mapper (left) and the submap building (right) on KITTI odometry sequences 00.

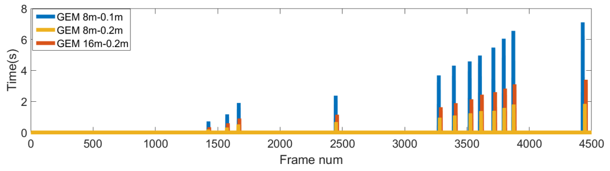


Fig. 7. Time evolution trend of GEM for the global dense map generation on KITTI odometry sequences 00.

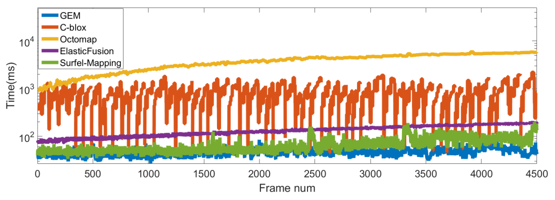


Fig. 8. Mapping time evolution trend of GEM, Octomap, C-blox, ElasticFusion and Surfel-Mapping on KITTI odometry sequences 00.

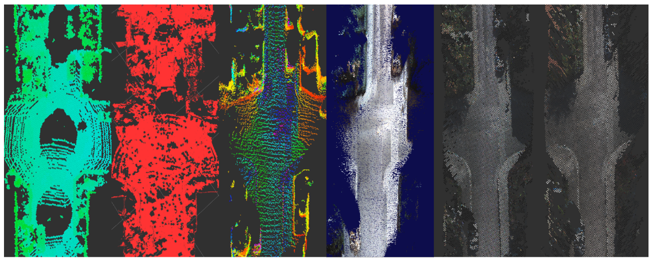


Fig. 9. The mapping results of the same region using Octomap, C-blox, Surfel-Mapping, ElasticFusion, GEM with LiDAR and stereo vision on KITTI dataset (left to right).

B. Real-time performance

The real-time performance is another important indicator of the mapping system supporting online navigation. As the global planner usually works by an external trigger, the local planner has a more critical requirement of real time. So we first evaluate the computation time of the local dense mapper on both desktop and embedded GPU. Then we compare the time with other comparative methods to show the evolution in a large-scale environment. In addition, we also qualitatively investigate the map density and dynamic objects response. When the system cannot achieve real-time performance, map density can be sparse, and cluttered by dynamic objects, incorrectly blocking the navigation.

Step-by-step time analysis. To evaluate the time complexity and the acceleration brought by GPU implementation, we evaluate the computation time of the local dense mapper into steps. The CPU based robocentric Elevation-mapping is utilized as a baseline [8], which has no annotation step. We run both methods on KITTI dataset for computation time collection. The local map size is $8\times8m^2$ with resolution $0.2m$. As shown in Tab. II, our approach achieves almost 20Hz on desktop and 6Hz on embedded TX2. By integrating the annotation into dynamic objects processing, our implementation is much faster than the original one that processing all grids. As a result, the most time consumable step in Elevation-mapping becomes the least time consumable step in GEM. The step taking most time in GEM is geometric map building due to the sequential processing when multiple points lying in the same grid.

Time evolution trend. To validate the scalability, we run all methods on the large-scale environment on the KITTI dataset. Refer to the architecture of GEM in Fig.2, we first compare the running time of local dense mapper and submap building (including local dense mapper time) of four reconstruction models with the local map size of $8 \times 8m^2$, $16 \times 16m^2$, and the resolution of $0.1m$ and $0.2m$ using lidar. As shown in Fig. 6, when the size of the local mapper becomes larger, more sensor data lies in the local map region, and more grids are allocated for fusion, resulting in more computation time. Considering that the grids in the map with $16 \times 16m^2$ and $0.2m$ resolution, and $8 \times 8m^2$ and $0.1m$ actually has 4 times more grids than that with $8 \times 8m^2$ and $0.2m$ resolution, the computation time only grows less than 2 times, which mainly owes to the GPU parallelization. In addition, fewer range measurements lying in the local region is also another reason. Besides, even the mapping environment becomes larger and larger, the time for local mapping keeps almost constant, which is important for real-time local planning. Accordingly, the submap building also demonstrates a similar time trend. In Fig. 7, time evolution trend for global dense map generation including global deformation and submap fusion is shown. The time increases with the growing mapping area since more submaps are created and processed to keep global consistency, leading to an area related with time complexity.

For fairness, we convert all points in the local map to the global map without considering their annotation, as comparative methods do not have this functionality. Due to compact and efficient mapping architecture with two threads, we can regard the submap building time as the running time of GEM. In contrast, we also evaluate the running time for Octomap, C-blox, ElasticFusion and SurfelMapping, of which the results are shown in Fig. 8. For Octomap, as all information is stored in the global volume, the evolution of the computation time becomes slower due to the growing mapping area. For C-blox, as there is the submap mechanism, the time for each submap keeps almost constant. But in each submap, there is a growing computation time with respect to the growing covered mapping area. When more submaps are built, processing time can be constrained. But data in each submap becomes less, leading to a small and sparse local map, which limits the horizon of planning. These two results show the advantage of decoupling the local map building and submap building in our method, thus achieving constant time for each step. For Surfel-Mapping, we see that similar time performance with ours is achieved, but there is also a growing trend with respect to the

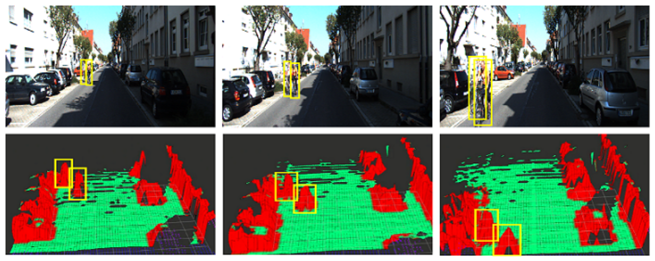


Fig. 10. The top row shows the scenes with cyclists highlighted in yellow boxes in frame 0645, 0650, 0654 of KITTI odometry sequences 00. The bottom row gives a series of local dense maps updated accordingly by GEM when two cyclists moving toward the vehicle.

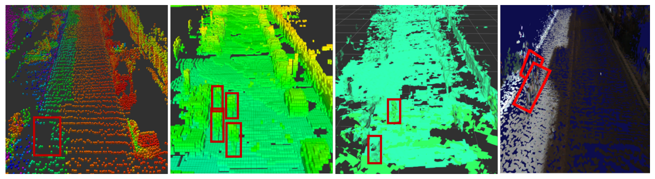


Fig. 11. The mapping results when dynamic cyclists moving toward the vehicle using Surfel-Mapping, Octomap, C-blox and ElasticFusion.

growing environment area. The main reason is the coupling of global mapping in local mapping. For ElasticFusion, it also has an ever-growing computational time. The main reason is that it keeps an ever-growing map and deformation graph in the memory. Also, the continuous surfel representation takes lots of memory.

Density of the map. Aiming at the on-the-ground navigation, we focus on the quality of terrain in the built map. We qualitatively compare the terrain in the local map by Octomap, C-blox, ElasticFusion, Surfel-Mapping and ours. The maps of the same area built by these methods are shown in Fig. 9. Dense maps built by Octomap and C-blox contains many holes. The main cause is the frames skipping due to the growing computation time for each step, which reflects the importance of keeping constant real-time performance. As GPU based implementation highly accelerates local mapping, the density of ElasticFusion outperforms CPU based methods. But it still has many small holes on the ground surface. These holes are regarded as obstacles in the planning stage, thus severely breaking the traversability of the terrain. For the Surfel-Mapping method, the map is represented by points, which calls for an additional step to render a dense map for navigation. The map built by our methods provides very dense terrain, satisfying the interface of the planner.

Dynamic objects response. Dynamic obstacles cannot be ignored for safe motion planning. Therefore, mapping the dynamic objects and remove them in time is an important indicator to evaluate a mapping system. We select a segment on KITTI where dynamic objects are presented to qualitatively compare the Octomap, C-blox, ElasticFusion, Surfel-Mapping and ours. Namely, two cyclists moving toward the vehicle from frame 0645 to frame 0655 in KITTI odometry sequences 00. In Fig. 10, our method annotates the cyclists as obstacles. We then update and clear them in the local dense map in time. The final states of the maps using other methods are shown in Fig. 11. Octomap and C-blox leave artifacts caused by the cyclists due to frame skipping. Surfel-Mapping directly

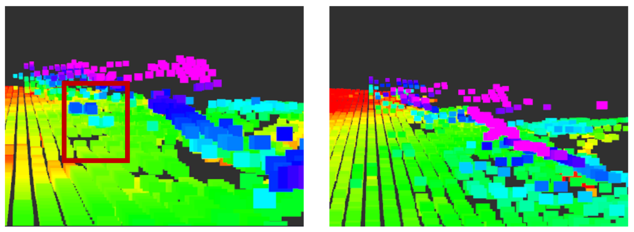


Fig. 13. The dynamic obstacles process of global map on the YQ dataset. Left image shows that a pedestrain highlighted by the red box is reserved in the global map. Right image shows that the pedestrain in the global map is cleared by the latest local map.

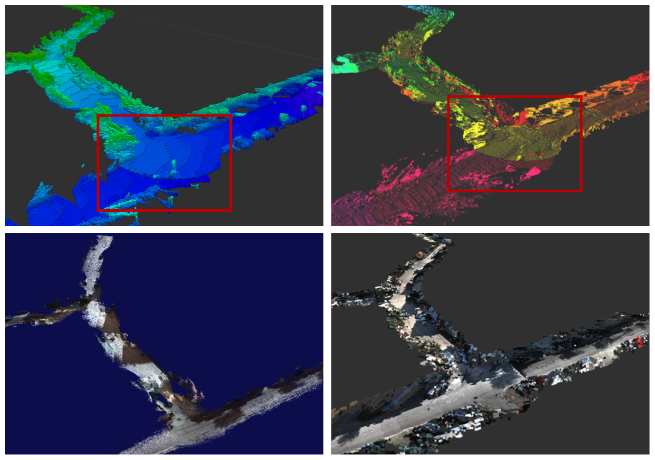


Fig. 14. The map built by Octomap (top left), C-blox (top right), ElasticFusion (bottom left) and GEM (bottom right) on Kitti dataset with stereo vision. The inconsist regions of Octomap and C-blox are highlighted.

filters the dynamic objects by dynamics checking. Note that even the objects are built in Surfel-Mapping, they cannot be cleared as no ray casting is conducted. This is the main reason that Surfel-Mapping is generally utilized as an offline mapping tool. For ElasticFusion, one can see that the cyclists cause the artifacts since it is difficult to remove all small surfels in continuous space using discrete rays, which reduces the traversable region for planning. These results demonstrate that fast GPU processing is not the only reason for dynamic objects updating. Another reason is that GEM has a specialized step of ray tracing to process the dynamic objects. Additionally, the ray casting only works in the local map to clear the dynamic objects. As for the global map which is aggregated from points in the local map, we have no observations to update even there are dynamic objects. However, one exception is that when loop closure happens, map maintenance is running to re-active the past submaps, thus the current local map can also update the past global map. As shown in Fig. 13, a pedestrian reserved in the global map can be cleared by the latest local map.

C. *Global map quality*

Global planning is triggered when the local path planning is blocked. Therefore the real-time requirement for global mapping is lower. Instead, the important indicator for global map is the consistency. We evaluate the global map consistency when correction from the SLAM system happens because of loop closure. In addition, the scalability is also a problem, as the

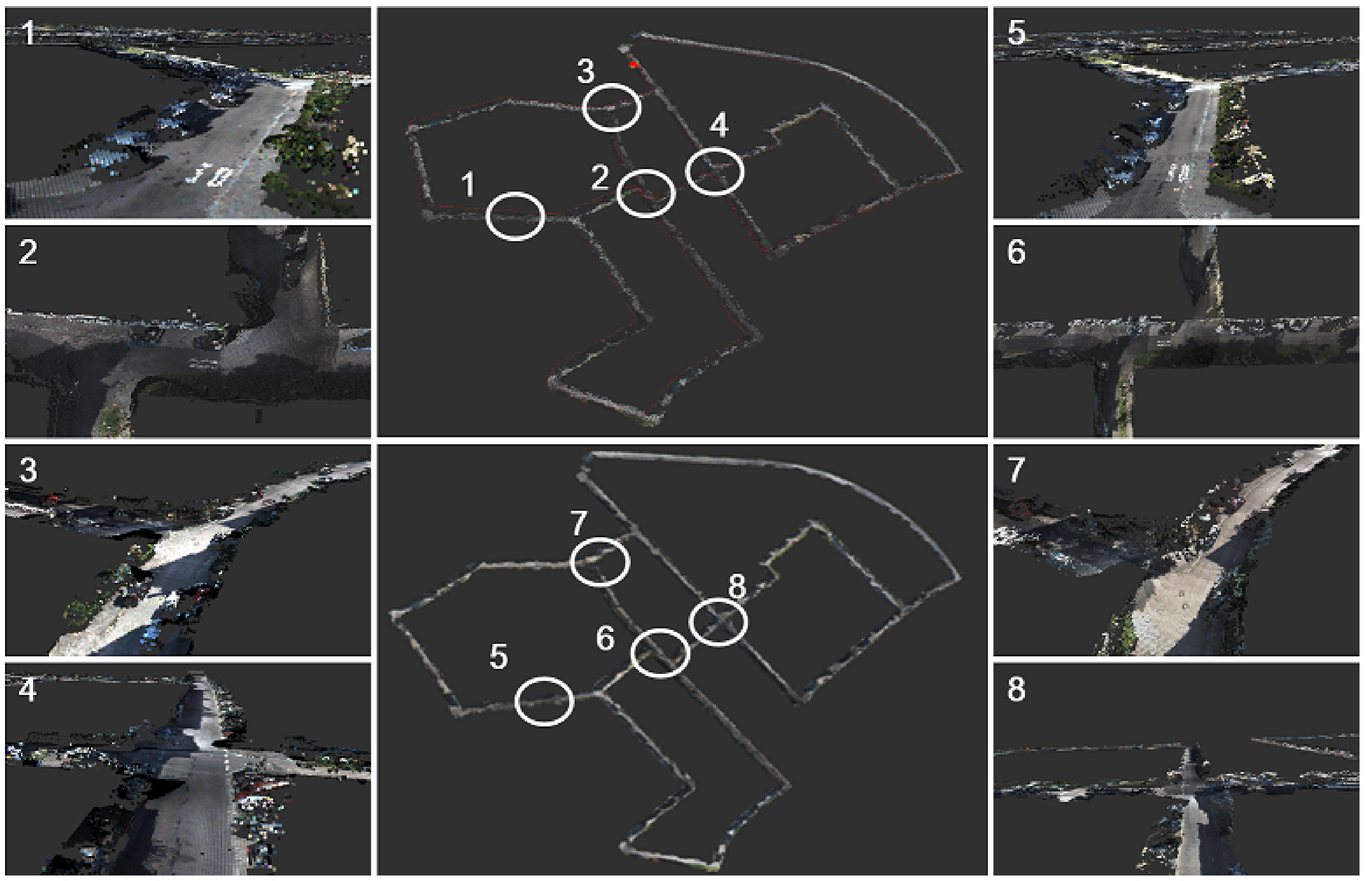


Fig. 12. The top image in the middle column shows the global dense map using LiDAR and the bottom one using stereo vision. The left column gives the details of the corresponding regions in the LiDAR built map, and the right column shows the details in stereo cameras built map.

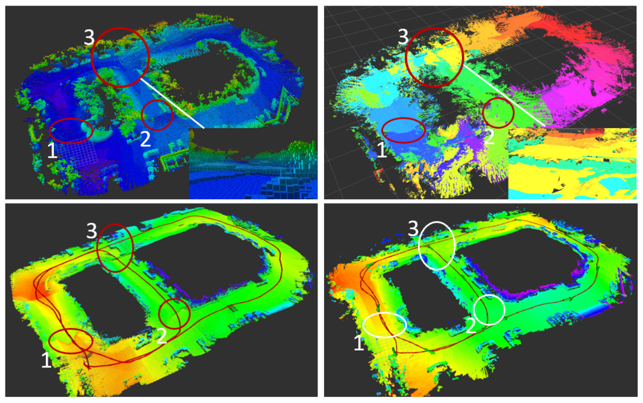


Fig. 15. The map built using Octomap (top left), C-blox (top right), GEM without (bottom left) and with (bottom right) global deformation on the YQ dataset. The inconsistency is highlighted in red circles.

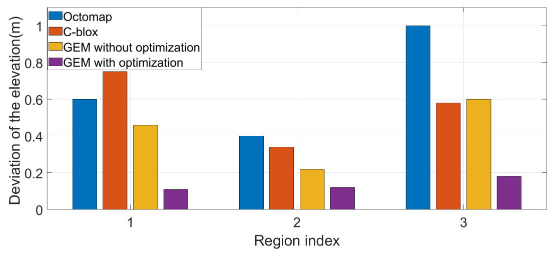


Fig. 16. The elevation smoothness of the local terrain with different methods on the YQ dataset. Region index indicates the region highlighted in Fig. 15.

robot operating in a bounded environment cannot have ever-growing computation. We demonstrate the evolution of storage to validate the scalability for globally consistent mapping methods. For fairness, ORB-SLAM2 is applied to all methods using vision sensors, and [38] is used as the LiDAR SLAM system for LiDAR-based methods.

Global consistency When mapping a large-scale environment, we cannot expect a globally consistent trajectory estimation without loop closure optimization. The global consistency of the mapping system focuses on the deformation mechanism. In Fig. 12, we show the global maps built by GEM for qualitative evaluation. To satisfy the dense repetitive observations for ElasticFusion and have a fair comparison, we use stereo vision to construct global maps by various methods in Fig. 14 and slow down package playback to maintain the performance of Octomap and C-blox. When looking into the regions where loop closure happens, we can clearly see the inconsistency in the map built by Octomap and C-blox since no deformation mechanism is equipped. While in the map built by GEM or ElasticFusion, there is no artifact in loop closure regions.

To test the mapping system in an off-the-road environment, we also run the methods using LiDAR on the YQ dataset, which has sloppy terrain. The map is configured with $12 \times 12m^2$ with resolution $0.2m$ in GEM. Note that ElasticFusion can't finish the complete map due to using sparse observations, so as a comparison, we build the maps by Octomap and C-blox on the YQ dataset in Fig. 15. The robot running off-the-

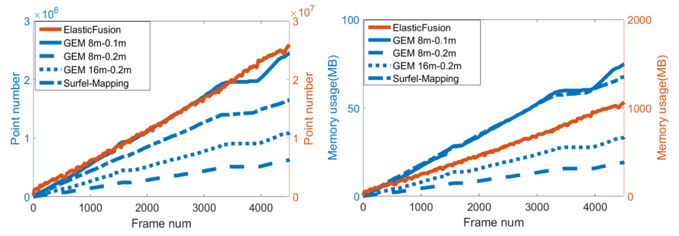


Fig. 17. The memory usage evolution trend of ElasticFusion, Surfel-Mapping and GEM with various local map sizes and resolutions on the KITTI dataset.

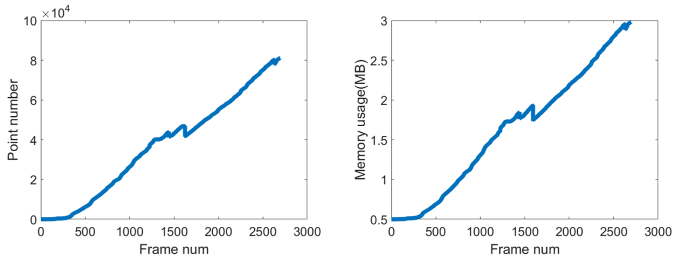


Fig. 18. The memory usage evolution trend of GEM with $12 \times 12m^2$ size and $0.2m$ resolution on the YQ dataset.

road on YQ is slower than the vehicle running on-the-road on KITTI, but the artifacts in loop closure regions still reflect the weakness of methods without deformation mechanism. The map built by GEM with and without global deformation is shown in Fig. 15. One can see that artifacts present when no global deformation is utilized, resulting in a similar map as Octomap and C-blox. To measure the accuracy of the global dense map, since it is almost impossible to acquire ground truth terrain model, we compare the elevation deviation in three regions where loop closures happen as highlighted in Fig. 15. We believe that the elevation smoothness of the local terrain can validate the value of map deformation given that the terrain is smooth in the real world. The results are shown in Fig. 16 that GEM with global deformation demonstrates the minimal deviation results as the corrected trajectory is used, which is more globally consistent than the others. A video of mapping on KITTI and YQ dataset is uploaded to [9].

Scalability In terms of memory storage, we record the number of stored points, which is highly related to total memory usage. Higher resolution and larger local map size increase the memory usage as shown in Fig. 17 on the KITTI dataset. Note that the mapping system detects multiple closed loops from frame 3300 to 4000. Correspondingly, the memory storage keeps invariant as the new observations are fused into previous submaps, which is the advantage brought by the map maintenance. Surfel-Mapping has memory storage related to the mapping area instead of trajectory length. It generates a similar trend to ours, validating the area-related memory complexity in our method. ElasticFusion has a total number of points up to 26 million, taking exponentially more memory than GEM and Surfel-Mapping, which is serious for robots' on-board processing. On the YQ dataset, there two loops with a segment of trajectory having loop closure. Accordingly, the memory usage from frame 1300 to 1600, and in the last segment on the YQ dataset reflects this fact as shown in Fig. 18, confirming the advantage of the map maintenance in GEM.

VIII. Conclusion

In this paper, we have presented a dense elevation system, GEM, which can build the local dense map in constant real-time performance, and the globally consistent dense map at the same time. The local map is built in a moving frame assigned to the robot, decoupling the process from the submap building and global map building. A CPU-GPU implementation is proposed to achieve geometric map building, traversability annotation and dynamic objects clearing, improving the mapping efficiency significantly. For global mapping, a collection of submaps is utilized as the representation, deforming the global map when the correction happens in the SLAM system to achieve global consistency. Finally, the accuracy, efficiency, consistency and scalability of GEM are validated on both simulation and real-world datasets for both on-the-road and off-the-road, demonstrating its feasibility for on-the-ground navigation.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[3] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, MIT Press (MA), 2014.

[4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[5] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[6] P. Pfaff, R. Triebel, and W. Burgard, "An efficient extension to elevation maps for outdoor terrain mapping and loop closing," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 217–230, 2007.

[7] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[8] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[9] "Gem: An online globally consistent elevation mapping system for on-the-ground navigation." https://www.youtube.com/watch?v=MufkLpkNhhY Accessed March, 2020.

[10] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[11] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[12] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, "Locnet: Global localization in 3d point clouds for mobile vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 728–733, IEEE, 2018.

[13] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

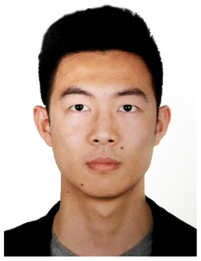
[14] D. Schlegel, M. Colosi, and G. Grisetti, "Proslam: Graph slam from a programmer's perspective," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, IEEE, 2018.

[15] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3144–3150, IEEE, 2019.

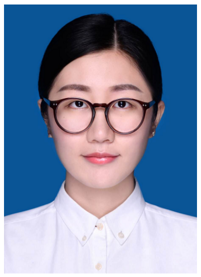
[16] I.-S. Kweon, M. Hebert, E. Krotkov, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *IEEE International Conference on Robotics and Automation*, pp. 997–1002, IEEE, 1989.

[17] I.-S. Kweon and T. Kanade, "High-resolution terrain map from multiple sensor data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 278–292, 1992.
[18] A. Kleiner and C. Dornhege, "Real-time localization and elevation mapping within urban search and rescue scenarios," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 723–745, 2007.
[19] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.
[20] 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 *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, IEEE, 2011.
[21] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, pp. 1–11, 2013.
[22] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.
[23] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.
[24] F. Gao, L. Wang, B. Zhou, L. Han, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *arXiv preprint arXiv:1907.00520*, 2019.
[25] D. Droeschel, M. Schwarz, and S. Behnke, "Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner," *Robotics and Autonomous Systems*, vol. 88, pp. 104–115, 2017.
[26] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Proceedings of Robotics: Science and Systems*, MIT Press(MA), 2015.
[27] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent tsdf-based dense mapping approach," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 995–1002, IEEE, 2018.
[28] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4475–4483, 2015.
[29] 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," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2175–2182, IEEE, 2018.
[30] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense surfel mapping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6919–6925, IEEE, 2019.
[31] B. Jia, J. Chen, K. Zhang, and Q. Wang, "Sequential monocular road detection by fusing appearance and geometric information," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 633–643, 2019.
[32] Z. Chen, X. Xu, Y. Wang, and R. Xiong, "Deep phase correlation for end-to-end heterogeneous sensor measurements matching," *arXiv preprint arXiv:2008.09474*, 2020.
[33] Z. Chen, X. Xu, J. Guo, Y. Wang, Y. Wang, and R. Xiong, "Collaborative localization of aerial and ground mobile robots through orthomosaic map," 2020.
[34] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3206–3211, IEEE, 2009.
[35] 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, 2013.
[36] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, 2012.
[37] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.
[38] X. Ding, Y. Wang, R. Xiong, D. Li, L. Tang, H. Yin, and L. Zhao, "Persistent stereo visual localization on cross-modal invariant map," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

Yiyuan Pan received BS from College of Information Engineering, Zhejiang University of Technology, Hangzhou, P.R. China in 2018. He is currently a MS candidate in Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. His latest research interests include dense mapping and robot perception.



Xuecheng Xu received BS from Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2019. He is currently a MS candidate in Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. His latest research interests include SLAM and air-ground collaborate localization.



Xiaqing Ding received BS from Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2016. She is currently a MS candidate in Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. Her latest research interests include SLAM and visual based localization.

Shoudong Huang received the Bachelor and Master degrees in Mathematics, Ph.D. in Automatic Control from Northeastern University, PR China in 1987, 1990, and 1998, respectively. He is currently an Associate Professor at Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. His research interests include nonlinear control systems and mobile robots simultaneous localization and mapping (SLAM), exploration and navigation.

Yue Wang received his PhD the from Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2016. He is currently an associate professor in the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. His latest research interests include mobile robotics and robot perception.

Rong Xiong received her PhD from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2009. She is currently a professor in the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. Her latest research interests include motion planning and SLAM.