Globally Consistent 3D LiDAR Mapping with GPU-accelerated GICP Matching Cost Factors

Kenji Koide¹, Masashi Yokozuka¹, Shuji Oishi¹, and Atsuhiko Banno¹

Abstract—This paper presents a real-time 3D LiDAR mapping framework based on global matching cost minimization. The proposed method constructs a factor graph that directly minimizes matching costs between frames over the entire map, unlike pose graph-based approaches that minimize errors in the pose space. For real-time global matching cost minimization, we use a voxel data association-based GICP matching cost factor that is able to fully leverage GPU parallel processing. The combination of the matching cost factor and GPU computation enables constraint of the relative pose between frames with a small overlap and creation of a densely connected factor graph. The mapping process is managed based on a voxel-based overlap metric that can quickly be evaluated on a GPU. We incorporate the proposed method with an external loop detection method in order to help the voxel-based matching cost factors to avoid convergence in a local solution. The experimental result on the KITTI dataset shows that the proposed approach improves the estimation accuracy of long trajectories.

Index Terms—3D LiDAR, SLAM, Mapping, GPU processing.

I. INTRODUCTION

ENVIRONMENTAL mapping is crucial for autonomous systems, and SLAM has been a major research topic in the robotics field. An important aspect of SLAM is global consistency. It is desirable that a mapping system is able to retain the consistency of every single part of a map after running on a long trajectory and closing large loops.

One way to refine a trajectory estimation result and improve the mapping consistency is pose graph optimization, which minimizes the relative pose errors between frames in the pose space. This approach has been well established in the literature and is widely used. Pose graph optimization requires modeling each relative pose constraint in the form of a Gaussian distribution (i.e., mean and covariance matrix). However, representing a relative pose, which is typically a result of scan matching, as a Gaussian distribution is clearly too approximated. Scan matching solutions have many local minima and thus cannot be accurately modeled in the form of a unimodal distribution. Furthermore, estimating the uncertainty (i.e., covariance matrix) of a scan matching result is difficult in practice. Most existing studies use only a constant covariance matrix, a simple weighting scheme, or Hessian-based closed-form covariance estimation, which tends to be optimistic. Inaccurate modeling of relative pose constraints can lead to deteriorated estimation accuracy of a long trajectory with large loops.

Global matching cost minimization is another approach to improve the consistency of a map. Early on, Lu and Milios proposed a graph-based 2D mapping approach that minimized the matching cost between frames over the entire map. This method was then extended to three dimensions by reducing the number of global optimization executions by explicitly hand-
dling loop closure events [8], [9]. These approaches ensured that all the frames were aligned together and thereby retained the local consistency of every part of the map (i.e., global consistency) while closing loops. They evaluated the global matching cost and updated each factor for every optimization iteration; this can be interpreted as the SE3 relative pose factor with variable mean and covariance. Furthermore, each factor can represent a deficient constraint in this way. They thus can more accurately model the constraint of the relative pose between frames compared to pose graph-based approaches. However, performing global matching cost minimization was still computationally expensive, and application to large-scale and real-time mapping was considered to be infeasible.

In this work, we revisit the global matching cost minimization approach with modern GPU computation techniques and propose a real-time and globally consistent 3D LiDAR mapping framework. The core of the proposed framework is the multi-scan registration algorithm, which minimizes the errors of Generalized ICP (GICP) matching cost factors with voxel-based data association [10], [11] over the entire map by fully leveraging GPU parallel processing. This approach has several advantages. First, this enables constraint of the relative pose between frames with a very small overlap, where it is difficult to explicitly estimate the relative pose through scan matching (Fig. 1(A)). Second, the GPU-powered implementation enables the creation of a massive amount of factors (Fig. 1(B)). Although we create a matching cost factor between every frame pair with an overlap rate larger than a small threshold (e.g., 2.5%), the global map optimization converges in a few seconds on a middle-class GPU.

The proposed framework consists of local and global mapping modules, which perform matching cost minimization locally and globally, respectively (see Fig. 2). Both of the mapping modules are managed based on a voxel-based overlap metric that can quickly be evaluated on a GPU. In order to prevent the voxel-based matching cost factors from becoming stuck at a local minimum, we explicitly detect a few loops with an external loop detector (e.g., ScanContext [12]) and add these loops to the factor graph as SE3 relative pose constraints. Through evaluation on the KITTI dataset [13], we show that the proposed approach improves the estimation accuracy of long trajectories with large loops.

The key contributions of this work are as follows:

1) We present a globally consistent 3D mapping framework based on the GPU-accelerated matching cost factor and show that the matching cost minimization over the entire map is feasible in real-time. To the best of our knowledge, this is the first real-time method that performs scan matching at a global scale.
2) We propose a mapping management mechanism based on an overlap metric that can quickly be evaluated on a GPU and enables the design of a general mapping process.
3) We show that the global matching cost minimization approach enables retention of the global consistency of large maps and increases the mapping quality.

II. RELATED WORK

Since the main contribution of this work is a method by which to retain the global consistency of a map, we focus in this section on global map optimization approaches.

A. Pose Graph Optimization

While many frontend methods for 3D LiDAR SLAM have been proposed [2], [5], [14], [15], most of these systems rely on pose graph-based maximum a posteriori estimation [1] as the backend in order to refine trajectory estimation results and improve the mapping consistency. Pose graph-based approaches construct a factor graph with relative pose (SE3) constraints and estimate the sensor trajectory that minimizes the errors in the pose space. This approach has been well established and has become the gold standard for the 3D LiDAR SLAM backend.

In pose graph optimization, relative pose constraints are modeled as a Gaussian distribution. However, the Gaussian distribution form is a too-approximated representation for scan matching results. A scan matching solution inherently has many local minima and thus cannot be accurately modeled in the unimodal distribution form, and this approximated representation would affect the optimization result once the scan matching converges to a local solution.

Furthermore, estimation of the covariance matrix of a scan matching result is difficult in practice [4]. Although closed-form uncertainty estimation methods based on the Hessian matrix of the cost function have been commonly used [16], [6], it is known that closed-form methods tend to be optimistic because these methods are not able to take into account the cost function deviation caused by data association changes [4]. On the other hand, Monte-Carlo-based covariance estimation methods can more accurately estimate the uncertainty of scan matching results [17]. These Monte-Carlo-based methods, however, are computationally expensive. Although data-driven covariance estimation approaches [4] have been proposed in order to balance the real-time performance and estimation accuracy, most existing SLAM frameworks use only a constant covariance matrix [2], a simple weighting scheme [5], [14], or Hessian-based closed form covariance estimation [6].

B. Deformation Graph

Map deformation is another approach to retain the surface consistency of mapping results including loops [18]. This approach constructs a graph that deforms the mapping result such that the local consistency is preserved. Deformation graph-based map-centric mapping approaches without estimation of the full sensor trajectory have been proposed [19]. These methods, however, do not accurately take all available information into account and may disrupt the global consistency of the map.

C. Bundle Adjustment

Bundle adjustment (BA) that simultaneously optimizes sensor poses and environmental parameters over frames has been important in the visual SLAM field [20]. It has been shown that
BA-based methods show good trajectory estimation and reconstruction accuracy while it is known to be computationally expensive. For real-time performance, BA is typically carried out at two different scale levels (real-time local BA and low frequency global BA) with a limited number of feature points \[20\]. Notably, Schöps et al. recently showed that direct BA-based visual odometry at a global level is feasible in real-time with GPU processing \[21\]. In the context of 3D LiDAR SLAM, however, it is rare to see BA-based approaches due to the difficulty of feature tracking on sparse LiDAR data, and few studies on BA-based approaches have been proposed \[22\], \[23\].

D. Global Matching Cost Minimization

Lu and Milios formulated the mapping problem as minimization of the matching cost of frames over a factor graph \[7\], and their method was extended to three dimensions by several studies by explicitly handling loop detection events \[8\], \[9\]. Their method was still considered to be infeasible in real-time because it needs to re-evaluate the matching cost of every frame pair at every optimization iteration.

Recently, Reijgwart et al. proposed a volumetric mapping method that takes into account registration errors between local submaps \[13\]. They used an efficient registration error metric based on Euclidean Signed Distance Field (ESDF) representation in order to avoid the costly correspondence search. The cost evaluation was, however, still computationally expensive, and optimization was carried out with only a random subset of residuals and with the support of SE3 relative pose constraints.

E. GPU-based SLAM

The GPU has been commonly used for almost every component of dense visual and RGB-D SLAM (from frontend \[21\] to backend \[18\]). In contrast, in the context of LiDAR SLAM, the use of GPU was mostly limited to accelerating scan matching in the frontend \[22\], \[24\]. While there have been also proposed deep learning-based frontend \[25\] and loop detection methods \[26\] with GPU processing, in most of the works, pose graph optimization performed on a CPU is in charge of global optimization.

III. METHODOLOGY

Figure 2 shows an overview of the proposed system. We first remove dynamic objects (e.g., cars and pedestrians) from input point clouds using RandLA-Net \[27\] and run an odometry estimation algorithm (e.g., MULLS \[15\]) to obtain an initial guess for the latest sensor pose. Meanwhile, we estimate the covariance matrix of each point from its k-neighboring points. Note that the costly nearest neighbor search is used only in this preprocessing step, which is performed once for every input point cloud.

The preprocessed point cloud and the sensor pose initial guess are fed to the local mapping module that merges approximately 10 to 20 frames into one submap, and the submaps are then merged into one global map in the following global mapping module. The core of the local and global mapping modules is the multi-scan registration algorithm that constructs a factor graph with voxelized GICP matching cost factors (see Fig. 3). This optimizes the sensor poses such that all neighboring frames are aligned together. In the submapping module, we construct a fully-connected factor graph. In the global-mapping module, we create constraints between the latest submap and every past submap that has a certain overlap with the latest submap. As a result, all of the submaps are aligned with not only adjacent submaps on the graph but also every revisited submap that results in closing loops implicitly. We obtain the final mapping result by concatenating submap point clouds based on the optimized trajectory.

A. Voxelized GICP Matching Cost Factor

We estimate a set of sensor poses \( T = \{ T_0, \cdots, T_i \} \) by minimizing the matching cost over a set of point cloud pairs \( M \). The objective function to be minimized is defined as:

\[
J^M(T) = \sum_{(i,j) \in P^M} c^M(p_i, p_j, T_i, T_j),
\]

where \( p_i \) and \( p_j \) are a point cloud pair, \( T_i \) and \( T_j \) are their poses, and \( c^M \) is a matching cost function.

As the matching cost function, we choose the voxelized GICP (VGICP) cost \[10\] that is as accurate as GICP and
suitable for GPU processing. The VGICP cost is based on the GICP distribution-to-distribution error that is the most accurate among the GICP distribution variants [11]. The GICP error between a point with covariance \( p_k = (\mu_k, C_k) \) and its corresponding point \( p'_k = (\mu'_k, C'_k) \) on a transformation \( T \) is defined as:

\[
e^{\text{GICP}}(p_k, T) = d_k^T(H + TC_kT^T)^{-1}d, (2)\]

where \( d_k = \mu'_k - T\mu_k \) is the residual between \( \mu_k \) and \( \mu'_k \).

In the original GICP algorithm, corresponding points are given by a nearest neighbor search, e.g., by a KD tree. However, the use of a KD tree is not suitable for a GPU because the KD tree uses a number of conditional branches, which affects the performance of the GPU. In order to maximize the processing speed, VGICP uses a voxel-based data association approach. It discretizes each input point cloud into voxels at resolution \( r \) and calculates the mean and covariance of each voxel based on the points that fall within the voxel. VGICP aggregates point distributions into one voxel distribution, unlike Normal Distributions Transform (NDT)-based algorithms that compute a voxel distribution from a set of points [28]. This approach enables a valid distribution on a voxel to be estimated from a set of points [28]. From the derivatives of Eq. 3, we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \).

\[
e^{\text{GICP}}(p_k, T) = d_k^T(H + TC_kT^T)^{-1}d, (2)\]

where \( d_k = \mu'_k - T\mu_k \) is the residual between \( \mu_k \) and \( \mu'_k \).

In order to manage the mapping process, we use criteria based on a simple fine-grained voxel-based overlap metric. We define the overlap rate between two point clouds \( P_i \) and \( P_j \) as the fraction of points \( p_k \in P_i \) that fall within a voxel of \( P_j \):

\[
s(p_k, P_j) = \begin{cases} 1 & \text{if } p_k \text{ fell in a voxel of } P_j \\ 0 & \text{otherwise} \end{cases} \]

overlapping \( P_i, P_j \) is defined as:

\[
e^M(P_i, P_j, T_i, T_j) = \sum_{p_k \in P_i} e^{\text{GICP}}(p_k, T_{ij}), (3)\]

where \( T_{ij} = T_i^{-1}T_j \) is the relative pose estimate between \( P_i \) and \( P_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \).

\[
e^{\text{GICP}}(p_k, T) = d_k^T(H + TC_kT^T)^{-1}d, (2)\]

where \( d_k = \mu'_k - T\mu_k \) is the residual between \( \mu_k \) and \( \mu'_k \).

In order to manage the mapping process, we use criteria based on a simple fine-grained voxel-based overlap metric. We define the overlap rate between two point clouds \( P_i \) and \( P_j \) as the fraction of points \( p_k \in P_i \) that fall within a voxel of \( P_j \):

\[
s(p_k, P_j) = \begin{cases} 1 & \text{if } p_k \text{ fell in a voxel of } P_j \\ 0 & \text{otherwise} \end{cases} \]

overlapping \( P_i, P_j \) is defined as:

\[
e^M(P_i, P_j, T_i, T_j) = \sum_{p_k \in P_i} e^{\text{GICP}}(p_k, T_{ij}), (3)\]

where \( T_{ij} = T_i^{-1}T_j \) is the relative pose estimate between \( P_i \) and \( P_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constrain \( T_i \) and \( T_j \). The corresponding points \( p'_k \) are given by looking up the voxel map of \( P_j \). From the derivatives of Eq. 3 we obtain a Hessian factor to constr

C. Global Mapping

The global mapping module takes the optimized submaps as input and optimizes their poses such that they are all aligned together. Every time a new submap is created, we compare the overlap between that submap and all past submaps, and create a matching cost factor between every submap pair with an overlap rate larger than a small threshold \( \theta_{\text{min}} \) (e.g., 10%). This results in a densely connected factor graph, as shown in Fig. 1. The proposed approach aggressively creates matching cost factors between submaps with a very small overlap, where scan matching would fail to align the submaps, and thus obtaining an accurate SE3 relative pose constraint is difficult (see Fig. 2). Although the matching cost factors over such submaps would represent deficient constraints, they do not disrupt the optimization because the entire system is well constrained by other factors. This approach helps in not only implicitly closing loops but also improving the odometry estimation accuracy because every submap is connected to all of the submaps in sight of that submap.
Fig. 4: Matching cost factor enables constraint of the relative pose between frames with a small overlap where scan matching can be deteriorated. The orange line in (A) indicates a matching cost factor between two frames, and (B) shows a corrupted scan matching result between these frames.

Fig. 6: Shifted Tukey’s biweight function.

IV. Evaluation

We evaluated the proposed framework on the KITTI odometry dataset [13]. We calculated the relative trajectory errors...
TABLE I: Average rotational relative trajectory errors (RTEs) [°/100m] on the KITTI dataset

| Sequence Num. | Loop closure | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | Mean ST | Mean ST/S |
|---------------|--------------|----|----|----|----|----|----|----|----|----|----|----|---------|-----------|
| 4541 | 1101 | 4661 | 801 | 271 | 2761 | 1101 | 1101 | 4071 | 1591 | 1201 | | |
| Proposed (matching cost) | ✓ | 0.16 | 0.10 | 0.12 | 0.19 | 0.10 | 0.10 | 0.07 | 0.11 | 0.18 | 0.11 | 0.15 | 0.14 | 0.13 | 0.15 |
| Proposed (matching cost) | ✓ | 0.12 | 0.09 | 0.10 | 0.19 | 0.10 | 0.06 | 0.08 | 0.10 | 0.14 | 0.08 | 0.15 | 0.11 | 0.11 | - |
| Proposed (SE3) | ✓ | 0.18 | 0.15 | 0.17 | 0.33 | 0.17 | 0.21 | 0.10 | 0.17 | 0.50 | 0.17 | 0.31 | 0.24 | 0.22 | - |
| LIDAM [29] | ✓ | 0.18 | 0.09 | 0.17 | 0.22 | 0.08 | 0.17 | 0.11 | 0.18 | 0.25 | 0.19 | - | 0.16 | 0.19 | - |
| MULLS [15] | ✓ | 0.13 | 0.09 | 0.13 | 0.22 | 0.08 | 0.07 | 0.08 | 0.11 | 0.17 | 0.12 | 0.19 | - | 0.13 | - |
| SuMa [9] | ✓ | 0.20 | 0.13 | 0.18 | 0.27 | 0.15 | 0.17 | 0.13 | 0.16 | 0.21 | 0.14 | 0.19 | - | 0.18 | 0.21 |
| ELO [24] | ✓ | 0.23 | 0.54 | 0.48 | 0.50 | 0.27 | 0.20 | 0.30 | 0.54 | 0.38 | 0.22 | 0.32 | - | 0.36 | 0.36 |
| IMLS-SLAM [30] | ✓ | 0.22 | 0.46 | 0.37 | 0.46 | 0.26 | 0.20 | 0.21 | 0.19 | 0.35 | 0.23 | 0.28 | - | 0.29 | 0.29 |
| SuMa++ [31] | ✓ | 0.28 | 0.46 | 0.32 | 0.48 | 0.52 | 0.25 | 0.34 | 0.32 | 0.29 | 0.40 | 0.47 | - | 0.33 | 0.38 |
| LiTAMING2 [14] | ✓ | 0.42 | 0.40 | 0.45 | 0.59 | 0.54 | 0.35 | 0.33 | 0.45 | 0.43 | 0.38 | 0.41 | - | 0.43 | - |
| Mean ST and S respectively indicate the means of sub-trajectory and sequence errors. |

Mean ST and S respectively indicate the means of sub-trajectory and sequence errors.

TABLE II: Average translational relative trajectory errors (RTEs) [m/100m] on the KITTI dataset

| Sequence Num. | Loop closure | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | Mean ST | Mean ST/S |
|---------------|--------------|----|----|----|----|----|----|----|----|----|----|----|---------|-----------|
| 4541 | 1101 | 4661 | 801 | 271 | 2761 | 1101 | 1101 | 4071 | 1591 | 1201 | | |
| Proposed (matching cost) | ✓ | 0.49 | 0.65 | 0.50 | 0.62 | 0.41 | 0.24 | 0.29 | 0.30 | 0.80 | 0.46 | 0.54 | 0.52 | 0.48 | 0.59 |
| Proposed (matching cost) | ✓ | 0.56 | 0.66 | 0.55 | 0.63 | 0.42 | 0.28 | 0.34 | 0.35 | 0.81 | 0.53 | 0.54 | 0.56 | 0.52 | - |
| Proposed (SE3) | ✓ | 0.58 | 0.61 | 0.60 | 0.69 | 0.44 | 0.38 | 0.34 | 0.37 | 1.51 | 0.68 | 0.74 | 0.74 | 0.63 | - |
| LOAM [29] | ✓ | 0.78 | 1.43 | 0.92 | 0.86 | 0.71 | 0.57 | 0.65 | 0.63 | 1.12 | 0.77 | 0.79 | - | 0.84 | 0.55 |
| MULLS [15] | ✓ | 0.51 | 0.62 | 0.55 | 0.61 | 0.35 | 0.28 | 0.24 | 0.29 | 0.80 | 0.49 | 0.61 | - | 0.49 | 0.65 |
| ELO [24] | ✓ | 0.54 | 0.62 | 0.69 | 0.61 | 0.35 | 0.28 | 0.29 | 0.27 | 0.83 | 0.51 | 0.61 | - | 0.50 | 0.68 |
| IMLS-SLAM [30] | ✓ | 0.50 | 0.82 | 0.53 | 0.68 | 0.33 | 0.32 | 0.33 | 0.33 | 0.80 | 0.55 | 0.53 | 0.55 | 0.52 | 0.69 |
| SuMa [9] | ✓ | 0.68 | 1.70 | 1.20 | 0.74 | 0.44 | 0.43 | 0.54 | 0.74 | 1.20 | 0.62 | 0.72 | 0.83 | 0.82 | 1.39 |
| SuMa++ [31] | ✓ | 0.64 | 1.60 | 1.00 | 0.67 | 0.37 | 0.40 | 0.46 | 0.34 | 1.10 | 0.47 | 0.66 | 0.70 | 0.70 | 1.06 |
| LiTAMING2 [14] | ✓ | 0.70 | 2.10 | 0.98 | 0.96 | 1.05 | 0.45 | 0.59 | 0.44 | 0.95 | 0.69 | 0.80 | 0.85 | 0.88 | - |
| LO-Net [25] | ✓ | 0.78 | 1.42 | 1.01 | 0.73 | 0.56 | 0.62 | 0.55 | 0.56 | 1.08 | 0.77 | 0.92 | - | 0.82 | - |

Mean ST and S respectively indicate the means of sub-trajectory and sequence errors.

(RTEs) averaged over 100 to 800 m trajectories with the KITTI official evaluation code (Development kit). We used an Intel Core i7-8700 (12 threads) with an NVIDIA GeForce GTX 1660 Ti to run the proposed framework. The parameters for the proposed framework used in the evaluation appear on the project page².

Comparison with State-of-the-Art Methods: We compared the proposed framework with state-of-the-art real-time 3D LiDAR SLAM methods (LOAM [32], MULLS [15], ELO [24], IMLS-SLAM [30], SuMa [2], SuMa++ [31], LiTAMING2 [14]), and a deep-learning-based method (LO-Net [25]).

We ran the proposed framework with two settings: 1) without implicit and explicit loop closure (i.e., every submap is connected to only other submaps in a sliding window) and 2) with both implicit and explicit loop closure. Similar to [15, 24, 30], we applied an intrinsic vertical scan angle correction to compensate for the point cloud distortion in the KITTI dataset for all the settings.

Tables I and II show the average rotational and translational RTEs, respectively, of the proposed method and the state-of-the-art methods. We noticed that while the KITTI official benchmark uses the average of sub-trajectory errors to summarize errors, several works report the mean of sequence errors that would overemphasize the errors of short sequences. For a fair comparison, we report both the metrics in Tables I and II (Means ST: mean of sub-trajectory errors, Mean S: mean of sequence errors).

The proposed method shows the best RTEs (0.14 / 0.13° and 0.52 / 0.48 m) without loop closing among the state-of-the-art methods for the sequence 00 to 10. In particular, the proposed method shows good accuracy on long trajectories (Sequences 00, 02, 05, and 08). For Sequence 11 to 21, the proposed method shows the RTEs that are ranked at the second place among LiDAR-based methods on the KITTI online leaderboard at the time of submission (0.15° and 0.59 m). With loop closing, although the rotational RTEs of the proposed method are largely improved, the translational RTEs

²See the project page for details: https://staff.aist.go.jp/k.koide/projects/ral2021/index.html

³The method GLIM on http://www.cvlibs.net/datasets/kitti/eval_odometry.php
Fig. 8: The color indicates per-point entropy (magnitude of inconsistency). SuMa exhibits inconsistent mapping results on the ground and walls due to inaccurate pose graph-based loop closing.

Fig. 9: Factor graph with SE3 relative pose factors. The line color indicates the magnitude of the error (Red: large error, Green: small error).

are slightly deteriorated (0.11 / 0.11° and 0.56 / 0.52 m). Similar trends are reported in several works [2], [15], and we infer point cloud distortion in the KITTI dataset affected the translational RTEs when loop closing is enabled.

To assess the mapping quality, we created a local map for every 10 frames by aggregating frames within 10 m and evaluated its mean map entropy (MME) [33]. For the sequence 00, the proposed method showed a small MME (0.14 ± 0.20) while a pose graph-based method, SuMa [2], exhibited a larger MME (0.19 ± 0.20)². Figure 8 shows local map MME of the proposed method and SuMa at a junction where a large loop closure happened. We can see points with large entropy (large inconsistency) on the ground and walls of the local map of SuMa, while the proposed method showed significantly smaller entropy (better consistency). This result suggests that the inaccurate modeling of the relative pose constraints in the traditional pose graph optimization can result in inconsistent mapping results while the global matching cost minimization approach can accurately retain the map consistency.

Ablation Study: In order to show that the matching cost minimization enables accurate trajectory estimation in comparison with pose graph optimization, we replaced every matching cost factor with an SE3 relative pose constraint estimated by GICP scan matching [11]. The initial guess for the scan matching is given based on the optimization result with the matching cost factors. The information matrix of each relative pose factor is calculated based on the Hessian matrix for the GICP scan matching result [16]. Considering that the scan matching would fail on small overlapping frames, we applied Huber’s robust kernel to each relative pose factor.

From Tables I and II, we can see that the accuracy of the proposed method strongly deteriorated with the relative pose factors, although the graph structure (submap connectivity) had not changed. Figure 9 shows a factor graph with SE3 relative pose factors. The color of lines indicates the magnitude of errors (Green: small error, Red: large error). We can see that factors between submaps in distance tend to have large errors because the scan matching failed to align the submap pairs with small overlap. The factors with large errors were removed by the robust kernel and thus did not contribute to the optimization result. Note that more factors would have worse relative pose measurements in a practical situation because a good initial guess cannot be expected for scan matching. This result suggests that the pose graph optimization scheme, which requires explicit estimation of the relative pose between frames, has difficulty in constraining distant frames and preserving the consistency over a long trajectory.

Runtime: Through the sequence 00, one of the longest sequences in KITTI, the proposed framework ran approximately twice as fast as the real-time elapsed (20 FPS). Note that we used pre-recorded frontend trajectory estimation results with MULLS [13] (ran at 26 FPS), and thus the processing time of the frontend algorithm was not taken into account.

Table III summarizes the runtime of the local and global mapping.

### Table III: Average processing time through KITTI 00

| Module          | Submodule | Time [ms]     |
|-----------------|-----------|---------------|
|                 |           | Local mapping |               |
| Factor creation | Optimization | 2.8 ± 5.0    |
|                 |           | 123.9 ± 130.4 |
|                 |           | Global mapping |             |
| Factor creation | ScanContext | 7.7 ± 12.3    |
|                 | Optimization | 7.1 ± 19.9    |
|                 |           | 884.0 ± 87.6  |

Fig. 10: Numbers of submaps and matching cost factors, as well as the global map optimization time for the KITTI 00 sequence.
mapping modules. The local submap optimization, which was performed approximately every 1.5 s, took 123.9 ms on average. The global optimization, which was performed approximately every 7.5 s, took 884.0 ms on average to optimize the factor graph, which had more than 4,500 factors at the end of the sequence by fully leveraging GPU parallel processing. Figure 10 shows how the runtime of the global optimization grew as the numbers of submaps and matching factors increased. Although a longer time (3 to 4 s) was required after closing large loops, most of the time, the optimization quickly converged in less than one second. Note that while the linearization and error evaluation of matching cost factors occupied most of the optimization time, the linear solver (performed on a CPU) took only approximately 3% of the total optimization time on average.

V. CONCLUSIONS
This paper presented a 3D LiDAR mapping framework based on VGICP matching cost factors. The GPU-accelerated matching cost evaluation enables simultaneous alignment of all frame pairs in the factor graph and preserves the global consistency over a long trajectory. The local and global mapping modules are managed based on the overlap metric, which can quickly be evaluated on a GPU, and the explicit loop closing mechanism helps the voxel-based matching cost factors to avoid convergence in a local minimum.

REFERENCES
[1] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based SLAM,” IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31–43, Dec. 2010.
[2] J. Behley and C. Stachniss, “Efficient surfel-based SLAM using 3D laser range data in urban environments,” in Robotics: Science and Systems XIV. Robotics: Science and Systems Foundation, June 2018.
[3] V. Reigwart, 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, vol. 5, no. 1, pp. 227–234, Jan. 2020.
[4] D. Landry, F. Pomerleau, and P. Giguere, “CELLO-3D: Estimating the covariance of ICP in the real world,” in IEEE International Conference on Robotics and Automation, IEEE, May 2019.
[5] T. Shan, B. Englott, D. Meyers, W. Wang, C. Ratti, and R. Daniela, “LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Oct. 2020, pp. 5135–5142.
[6] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D LIDAR SLAM,” in IEEE International Conference on Robotics and Automation, IEEE, May 2016.
[7] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” Autonomous Robots, vol. 4, no. 4, pp. 333–349, Oct. 1997.
[8] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, “Globally consistent 3D mapping with scan matching,” Robotics and Autonomous Systems, vol. 56, no. 2, pp. 130–142, Feb. 2008.
[9] J. Sprickerhoff, A. Nüchter, K. Lingemann, and J. Hertzberg, “A heuristic loop closing technique for large-scale 6D SLAM,” Automatica, vol. 52, no. 3, pp. 199–222, Jan. 2011.
[10] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, “Voxelized GICP for fast and accurate 3D point cloud registration,” in IEEE International Conference on Robotics and Automation, IEEE, May 2021.
[11] A. Segal, D. Haehnel, and S. Thrun, “Generalized-ICP,” in Robotics: Science and Systems V. Robotics: Science and Systems Foundation, June 2009.
[12] G. Kim and A. Kim, “Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Oct. 2018.
[13] A. Geiger, P.Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2012.
[14] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, “LiFAMIN2: Ultra light lidar-based slam using geometric approximation applied with KL-divergence,” in IEEE International Conference on Robotics and Automation, IEEE, May 2021.
[15] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, “MULLS: Versatile lidar slam via multi-metric linear least square,” in IEEE International Conference on Robotics and Automation, IEEE, May 2021.
[16] O. Bengtsson and A.-J. Baerveldt, “Robot localization based on scan-matching—estimating the covariance matrix for the IDC algorithm,” Robotics and Autonomous Systems, vol. 44, no. 1, pp. 29–40, July 2003.
[17] T. M. Iversen, A. G. Buch, and D. Kraft, “Prediction of ICP pose uncertainties using monte carlo simulation with synthetic depth images,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Sept. 2017.
[18] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, “Deformation-based loop closure for large scale dense RGB-D SLAM,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Nov. 2013.
[19] C. Park, P. Moghadam, S. Kim, A.Elfes, C. Fookes, and S. Sridharan, “Elastic LiDAR fusion: Dense map-centric continuous-time SLAM,” in IEEE International Conference on Robotics and Automation, IEEE, May 2018.
[20] R. Mur-Artal and J. D. Tardos, “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, Oct. 2017.
[21] T. Schops, T. Sattler, and M. Pollefeys, “BAD SLAM: Bundle adjusted direct RGB-D SLAM,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2019.
[22] Z. Liu and F. Zhang, “BALM: Bundle adjustment for lidar mapping,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3184–3191, Apr. 2021.
[23] D. Wisth, M. Camurri, S. Das, and M. Fallon, “Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1004–1011, Apr. 2021.
[24] X. Zheng and J. Zhu, “Efficient LiDAR odometry for autonomous driving,” IEEE Robotics and Automation Letters, pp. 1–1, 2021.
[25] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, “LO-net: Deep real-time lidar odometry,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2019.
[26] X. Chen, T. Labe, A. Milioto, T. Röhlung, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, “OverlapNet: Efficient semantic segmentation of large-scale point clouds,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2020.
[27] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, “Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations,” International Journal of Robotics Research, vol. 31, no. 12, pp. 1377–1393, Sept. 2012.
[28] J. Zhang and S. Singh, “LOAM: Lidar odometry and mapping in real-time,” in Robotics: Science and Systems X. Robotics: Science and Systems Foundation, July 2020.
[29] T. M. Iversen, A. G. Buch, and D. Kraft, “Prediction of ICP pose uncertainties using monte carlo simulation with synthetic depth images,” in IEEE/RSJ International Conference on Robotics and Automation, IEEE, Sept. 2015.
[30] J.-E. Deschaud, “IMLS-SLAM: Scan-to-model matching based on 3D laser scans,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Oct. 2018.
[31] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, “Elastic LiDAR fusion: Dense map-centric continuous-time SLAM,” in IEEE International Conference on Robotics and Automation, IEEE, May 2018.
[32] R. Mur-Artal and J. D. Tardos, “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, Oct. 2017.
[33] T. Schops, T. Sattler, and M. Pollefeys, “BAD SLAM: Bundle adjusted direct RGB-D SLAM,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, June 2019.
[34] Z. Liu and F. Zhang, “BALM: Bundle adjustment for lidar mapping,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3184–3191, Apr. 2021.
[35] D. Wisth, M. Camurri, S. Das, and M. Fallon, “Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1004–1011, 2021.