

# LT-mapper: A Modular Framework for LiDAR-based Lifelong Mapping

Giseop Kim<sup>1</sup> and Ayoung Kim<sup>2\*</sup>

**Abstract**—Long-term 3D map management is a fundamental capability required by a robot to reliably navigate in the non-stationary real-world. This paper develops open-source, modular, and readily available LiDAR-based lifelong mapping for urban sites. This is achieved by dividing the problem into successive subproblems: multi-session SLAM (MSS), high/low dynamic change detection, and positive/negative change management. The proposed method leverages MSS and minimizes potential trajectory error; thus, a manual or good initial alignment is not required for change detection. Our change management scheme preserves efficacy in both memory and computation costs, providing automatic object segregation from a large-scale point cloud map. We verify the framework’s reliability and applicability even under permanent year-level variation, through extensive real-world experiments with multiple temporal gaps (from day to year).

## I. INTRODUCTION

During long-term mapping using light detection and ranging (LiDAR) sensor, we encounter changes in an environment as in Fig. 1. The perceived snapshot of the environment contains both ephemeral and persistent objects that may change over time. To handle this change properly, long-term mapping must be solved for autonomous map maintenance [1] by detecting, updating, and managing the environmental changes accordingly. In doing so, the challenges in scalability, potential misalignment error, and map storage efficiency should be addressed and resolved toward lifelong map maintenance.

1) *Integration to multi-session SLAM for scalability*: Some studies regarded change detection as a post-process of comparing multiple pre-built maps associated with temporally distant and independent sessions. As reported in [2], alignment of multiple sessions in a single global coordinate may severely limit scalability. Following their philosophy, in this work, we integrate multi-session SLAM (MSS) and align sessions with anchor nodes [2] to perform change detection in a large-scale urban environment beyond a small-sized room [3]. Our framework consists of a LiDAR-based multi-session 3D simultaneous localization and mapping (SLAM) module, named **LT-SLAM**.

2) *Change detection under SLAM error*: Change detection between two maps would be trivial if maps were perfectly aligned. Early works [5, 3, 6, 7] in map change detection relied on the strong assumption of globally well-aligned maps

<sup>1</sup>G. Kim is affiliated with Autonomous Driving Group, NAVER LABS, Seongnam-si, Gyeonggi-do, S. Korea giseop.kim@naverlabs.com

<sup>2</sup>A. Kim is with the Department of Mechanical Engineering, SNU, Seoul, S. Korea ayoungk@snu.ac.kr

\*This work was supported by the NRF (NRF-2019K2A9A1A06070173) and MOLIT (No.22SMIP-A158708-03).

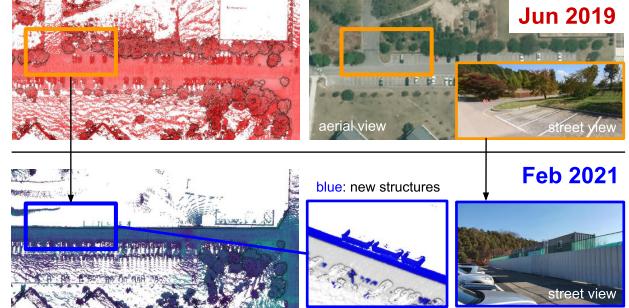


Fig. 1: An example of permanent structural changes over  $\sim 1.5$ -year temporal gap. (Top) KAIST 01 of MuRan dataset [4] (June 2019). (Bottom) KAIST 04, recently released in extended sequences (February 2021). A construction wall appeared over time; the previously existing parking spaces and trees disappeared. **LT-mapper** can accurately register the temporally disjointed maps and detect pointwise changes (e.g., blue points in the bottom blue box).

with no error and avoided handling this ambiguity issue. Unfortunately, trajectory error inevitably occur in reality. We reconcile this potential misalignment during our change detection, and enable the proposed method to handle potential alignment error robustly. To deal with the ambiguity, we propose a scan-to-map scheme with projective visibility, using range images of multiple window sizes named **LT-removert**. By extending an intra-session change detection method [8], the LT-removert includes both intra-/inter-session change detection, thereby further segregating high and low-dynamic objects [5] from the change.

3) *Compact place management*: In addition to change detection, we present and prove a concept of change composition. Once the change is detected, the decision for map maintenance should be followed to determine what to include or exclude. Using this feature, ours not only maintains an up-to-date map such as existing works [1, 3], but also extracts stable structures with higher *placeness*; thereby, we construct a reliable 3D map with authentically meaningful structures for other missions, such as cross-modal localization [9] and long-term localization [10]. This final module, named **LT-map**, manages the changes and enables a central map to evolve in a place-wise manner.

In sum, we propose a novel modular framework for LiDAR-based lifelong mapping, named **LT-mapper**. Each module in the framework can run separately via file-based in/out protocol. Unified and modular lifelong mapping has barely been made for 3D LiDAR, unlike recently (but partially) delivered visual-based methods [11, 12, 13, 14, 15]. To the best of our knowledge, LT-mapper is the first open modular framework that supports LiDAR-based lifelong mapping in complex urban sites. The proposed has the following

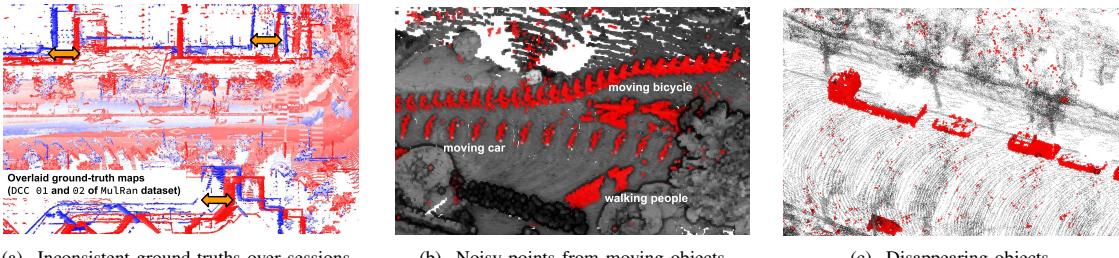


Fig. 2: Three challenges for LiDAR-based lifelong mapping in urban sites. (a) Overlaid ground-truth (GT) maps of MuRan [4] dataset DCC 01 and 02. Even with highly accurate sensors (e.g., RTK-GPS), GT maps may not be globally consistent along the temporal axis. (b) Moving objects (red dots) linger on a map as ghost points. (c) Structures or objects may disappear (red dots) or appear anew at a different session. These changes should be updated properly.

contributions:

- We integrate MSS with change detection and handle sessions resiliently via anchor node. The submodule **LT-SLAM** can stitch multiple sessions in a shared frame using only LiDAR.
- The submodule **LT-removert** overcomes alignment ambiguity between sessions with remove-then-revert algorithm along spatial and temporal axes.
- The submodule **LT-map** can produce both an up-to-date map (*live map*) and a persistent map (*meta map*) efficiently, while storing changes as a *delta map*. By exploiting delta maps, restoration and change detection become memory and computation cost-effective.
- The aforementioned modules are packaged within a single framework, and it is publicly released<sup>1</sup> with readily available console-based commands. Also, we provide real-world experiments with multiple temporal gaps (day to year).

## II. RELATED WORKS

1) *Multi-session SLAM*: In [1, 3], a query scan is assumed to be well localized within the map. However, in the real-world outdoor environment, SLAM errors exist and registration between scans may be vulnerable, failing even with small and partial structural variance. Thus, as claimed in [2, 14], jointly smoothing the multiple sessions can improve query-to-map localization performance despite potential motion drifts [5].

2) *3D Change Detection*: Given well-aligned maps, set union and difference operations can be conducted for extracting map-to-map complements [3, 6]. Otherwise, visibility-based scan-to-map discrepancy comparison [1, 7, 8, 16] has been a popular choice, because of the small covisible volume and inherent localization errors. Removert [8] leveraged range images of multiple window sizes. However, it was restricted to a single session and has not treated high and low dynamic points separately.

3) *Lifelong Map Management*: Lifelong map management should consider two factors: 1) which entity (representation), 2) how to be updated (update unit)?

**Representation.** The long-term map representation varies from traditional occupancy grid maps [17, 18, 19] to frequency domain representation [20]. For change detection in

3D environments, dealing with a direct raw 3D point cloud may be preferred over the occupancy map-based ones.

**Update Unit.** With respect to the atomic map update unit, we manage changes at a keyframe level. This contributes to systems scalability, without being restricted to a fixed global frame [1] or room level [3].

4) *Modular Design of Lifelong Mapper*: The abovementioned exiting modules have been developed individually, whereas a unified system has hardly been discovered for LiDAR. DPG-SLAM [5] combined the full modules but was constrained in SE(2) space and lack of 3D change detection. [1] was also equipped with entire modules, except for MSS.

## III. OVERVIEW

LT-mapper is fully modular and supports the three aforementioned functionalities. The overall pipeline is composed of three modules (Fig. 3), which run sequentially and independently. Unlike existing LiDAR-based change detection [21] equipped with expensive localization suites, our system requires only a single LiDAR sensor (optionally IMU for odometry at the initial pose-graph construction).

Accurate alignment between temporarily disconnected sessions is elusive in real-world outdoor environments, as can be seen in Fig. 2(a). In LT-SLAM module, we utilize multi-session SLAM that jointly optimizes multiple sessions accompanied with robust inter-session loop detection from a LiDAR-based global localizer. In this module, a query measurement is registered to the existing central (target) map.

We also need to consider measurements' spatial volatility. For example, in Fig. 2(b), a contracted point cloud map may be noisy, due to surrounding moving objects (red dots) even with the accurate odometry [8, 22]. These volatile objects contribute less to a place's distinctiveness than stationary points. Thus, these *high dynamic (HD)* points should be pre-removed before the between-session-differences calculation

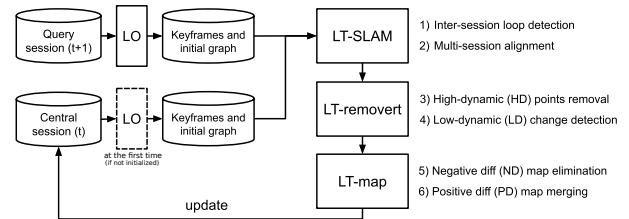


Fig. 3: A modular pipeline of **LT-mapper** system. The framework is composed of three modules: **LT-SLAM**, **LT-removert**, and **LT-map**.

<sup>1</sup>The code is available at <https://github.com/gisbi-kim/lt-mapper>.

in LT-removert module.

After aligning a query and a central (target) session and removing the HD points, we detect changes by applying set difference operation between query measurements and a central map, as in Fig. 2(c). We call the change *low dynamic (LD)*, and it is further divided into two classes: newly appeared points (*positive difference (PD)*) and disappeared points (*negative difference (ND)*).

```
# Read single-session graphs and their keyframes
./ltslam      # with params_ltslam.yaml
# Save aligned graphs

# Read the aligned graphs and keyframes' submaps
./ltremovert   # with params_ltremovert.yaml
# Save PD, ND keyframe scans

# Read the PD, ND keyframe' submaps
./ltmap        # with params_ltmap.yaml
# Save updated submaps [and merged maps for viz]
```

#### IV. LT-MAPPER

In this section, we give details of the three modules of LT-mapper. We define a session  $\mathcal{S}$  as

$$\mathcal{S} := (\mathcal{G}, \{(\mathcal{P}_i, d_i)\}_{i=1,\dots,n}), \quad (1)$$

where  $\mathcal{G}$  is a pose-graph text file (e.g., .g2o format) containing a set of pose nodes' indexes and initial values, odometry edges, and optionally putative intra-session loop edges. This initial pose-graph can be constructed by using any existing LiDAR (-inertial) odometry algorithms [23, 24, 25, 26, 27]. We allow potential navigational drifts and overcome the intra-session drifts via multi-session pose-graph optimization. The  $(\mathcal{P}_i, d_i)$  are a 3D point cloud  $\mathcal{P}$  and the its global descriptor  $d$  (e.g., [28, 10, 29, 30, 31]) for the  $i^{\text{th}}$  keyframe. We assign an equidistant sampled keyframes and  $n$  is the number of total keyframes.

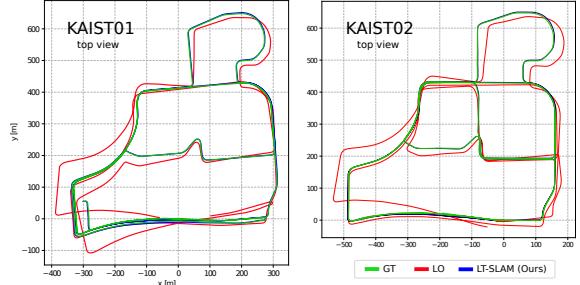
##### A. LT-SLAM: A Multi-session SLAM Engine

We denote the existing session  $\mathcal{S}_{t_c}$  at time  $t_c$  as *central* ( $C$ ), and the newly obtained session  $\mathcal{S}_{t_q} > t_c$  as *query* ( $Q$ ). Given a pair of the central and query sessions, LT-SLAM aligns the two sessions.

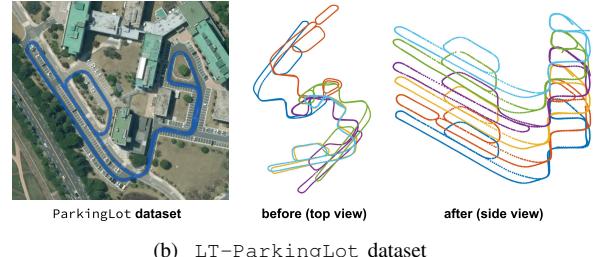
The incoming sessions' pose-graphs preserve their own coordinates and LT-SLAM utilizes the anchor node-based inter-session loop factors [2, 32, 33]. As Kim et al. [2] reported, the anchor node can successfully estimate a between-session offset, resolving their intra-session drifts. The anchor node-based loop factor for a relative pose measurement  $z$  is

$$\begin{aligned} & \phi(\mathbf{x}_{C,i}, \mathbf{x}_{Q,j}, \Delta_C, \Delta_Q) \\ & \propto \exp\left(-\frac{1}{2}\|((\Delta_C \oplus \mathbf{x}_{C,i}) \ominus (\Delta_Q \oplus \mathbf{x}_{Q,j})) - z\|_{\Sigma_z}^2\right), \end{aligned} \quad (2)$$

where  $\mathbf{x}$  means a SE(3) pose,  $i$  and  $j$  are pose variable indexes.  $\oplus$  and  $\ominus$  are the SE(3) pose composition operators [34].  $\Delta$  indicates an anchor node, which is also a SE(3) pose variable. The central session's anchor node  $\Delta_C$  has very small covariance while the query's  $\Delta_Q$  has a very large value.



(a) KAIST sequences of MulRan dataset



(b) LT-ParkingLot dataset

Fig. 4: (a) The baseline LO (red) is obtained by LIO-SAM's [26] odometry without loop closures. Scan Context-based inter-session loops successfully mitigated each session's internal drifts. (b) The left is an aerial view of LT-ParkingLot dataset. The middle plot shows different sessions' trajectories acquired at different dates. While covering the same area, trajectories were not aligned, as they occupy their own coordinates. The right plot shows that LT-SLAM simultaneously estimates between-session offsets while reducing the drifts. Thus, multiple sessions can be aligned in a shared world coordinate for change detection. The height differences are imposed for clear visualization.

We need to identify a loop-closure candidate  $(i, j)$  between sessions  $C$  and  $Q$ . For robust inter-session loop detection, we adopt Scan Context (SC) [10, 35] due to their long-term global localization capability and light computation cost. After the inter-session loop is detected, a 6D relative constraint between two keyframes is calculated via Iterated Closest Point (ICP) using their submap point clouds  $\mathcal{P}_{C,i}$  and  $\mathcal{P}_{Q,j}$ . We only accept loops with acceptably low ICP's fitness scores, and use the score for an adaptive covariance  $\Sigma_z$  in (2). We also use robust back-end (e.g., [36, 37]) for all inter-session loop factors for safe optimization under inevitable false loop detections. Given the initially aligned sessions using SC-loops, we further refine the graph using radius search loop detection (i.e., based on pose proximity) for non-SC-detected keyframes to finely stitch the sessions.

Finally, each session's trajectory is optimized within their own coordinates (denoted  ${}^C\mathcal{G}_C^*$  and  ${}^Q\mathcal{G}_Q^*$ ) as in Fig. 4(a). The optimized maps are then represented in a shared world coordinate  $W$  to be consumed by LT-removert introduced in §IV-B. To do so, LT-SLAM returns pose-graphs  ${}^W\mathcal{G}_C^*$  and  ${}^W\mathcal{G}_Q^*$  by applying the below transforms for each pose  $\mathbf{x}$  in a graph:

$${}^W\mathbf{x}_C^* = \Delta_C^* \oplus {}^C\mathbf{x}_C^* \text{ and } {}^W\mathbf{x}_Q^* = \Delta_Q^* \oplus {}^Q\mathbf{x}_Q^*. \quad (3)$$

The right in Fig. 4(b) shows the aligned multiple sessions sharing the same coordinates.

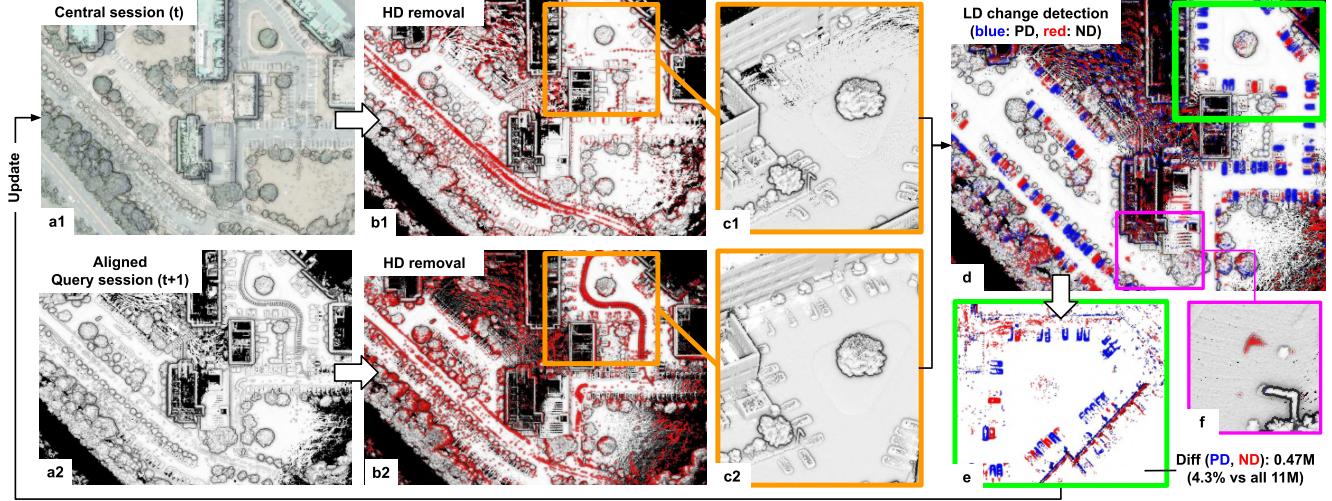


Fig. 5: A visualization of **LT-removert** pipeline. (a) LT-removert receives aligned central and query maps from LT-SLAM. (b, c) cleaned maps with HD points removed. (d, e) LD change detection (i.e., PD and ND segmentation). (f) deletion of unremoved HD points via multi-sessions.

### B. LT-removert: Two-session Change Detection

As mentioned in §III, the dynamic points are classified into HD and LD. In our second module, LT-removert, we first remove HD without erasing LD points. We denote  $LD_C^Q$  means 3D points that are low dynamic changes detected at a place (keyframe) between the session  $C$  (from) and  $Q$  (to).

1) *High Dynamic (HD) Points Removal*: We choose Removert [8] for our HD points removal engine. Using range image-based discrepancy, Removert utilizes different sizes of windows to alleviate the pose ambiguity. For example, Fig. 5 (b) and (c) show before and after of applying Removert.

2) *Low Dynamic (LD) Change Detection*: Once two sessions are aligned and HD points within them were removed, we compare query and central sessions to parse LD points. To do so, we construct a kd-tree for the target map and test whether a source map’s point has  $k$  target map points within a threshold  $r$  m (if not, the point is LD). Then, the ND and PD points are parsed.

3) *Weak ND Preservation (Handling Occlusions)*: Another critical factor to consider is occlusion as argued in [3]. In Fig. 6, an example is given to show the effect of occlusion in determining valid PD and ND. Naturally, the central session A will be compared against the query session B (case 1). However, let us consider the reversed case (case 2) when B occurs prior to A. In this case, some ground points were occluded by walls and became ND points. However, these ground points should not be removed. We name them as *weak ND* and examine further segregation to avoid falsely removing occluded static points.

For this step, we again employ Removert but with modification. Unlike the original Removert, which removes near map points, the modified Removert removes further points in the raw ND map and reverts them to the static map. The bottom right in Fig. 6 shows the preserved weak ND points (gray) being correctly reverted to the static map.

4) *Strong PD for Meta-map Construction*: We can consider a similar strong/weak classification for PD that is related to whether it retains permanent static structures. We

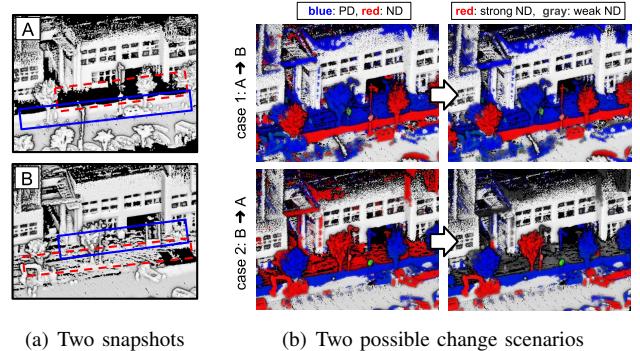


Fig. 6: (a) The exemplars (A) and (B) from DCC 01 and 02 of MulRan dataset. (A) includes occlusion where no points exist (red dotted box), due to the construction wall (blue-lined box). The wall disappears later, revealing the ground points in (B). (b) Selection of central session as A and query session as B will be Case 1; the reversed case is Case 2. The left column is the naive ND and PD points from the set difference operation. The right column shows accepted ND and PD points after weak ND point validation. In Case 1, red ND points were mostly removed while adding entire blue PD points. In Case 2, some ground points are occluded and falsely marked as ND in the left column. In the right column, only strong ND (red) points are removed, and weak ND points are reverted (i.e., ground points marked with gray).

call strong PD for the points spatially behind. If we only retain strong PD, as in Fig. 7, we can construct a map with maximum volume by carving out the space conservatively. In that sense, we can construct two types of static maps: *meta map* by removing weak PD and *live map* by retaining weak PD. The examples of *meta map* and *live map* are drawn in Fig. 9.

### C. LT-map: Map Update and Long-term Map Management

Given the detected LD, LT-map performs a between-session change update for each keyframe of the central session. The between-session change composition operator  $\odot$  is defined as:

$$\mathcal{P}_C = \tilde{\mathcal{P}}_C \odot LD_C^Q \triangleq \tilde{\mathcal{P}}_C - ND(LD_C^Q) + PD(LD_C^Q), \quad (4)$$

where  $\tilde{\mathcal{P}}_C$  is a keyframe’s HD removed point cloud. The function  $ND(\cdot)$  and  $PD(\cdot)$  return ND and PD points near

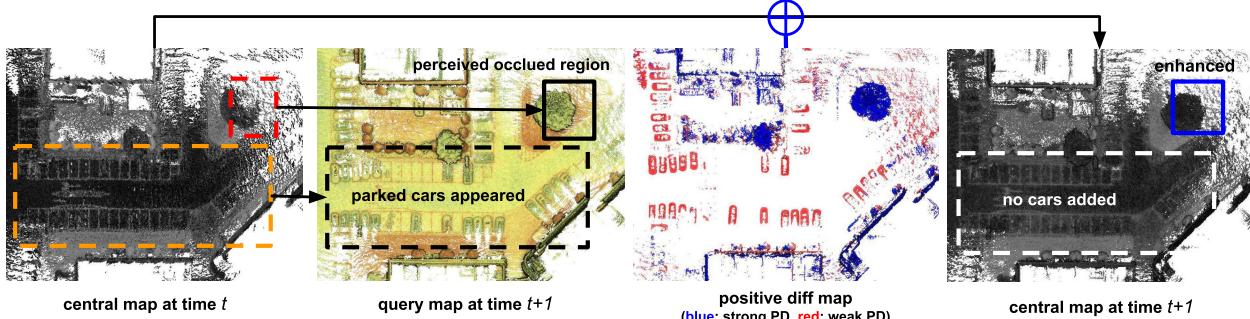


Fig. 7: The sample scene from LT-ParkingLot dataset. Strong PD includes newly captured permanent structures; weak PD tends to be short-term stationary or periodic changes (e.g., parked cars).

the keyframe and represented in the keyframe’s local coordinate. The  $-$  and  $+$  are set difference and union operation on 3D point space. This *delta map* containing only differences benefits compared to the snapshot-based methods that up/download the whole map. For example, in Fig. 5, transmitting the whole new map to a server requires 11 M points, whereas only 0.47 M points are needed when using *delta maps* (only 4.3 % of the entire map).

## V. EXPERIMENTAL RESULTS

### A. Implementation Detail and Dataset

**1) Implementation Detail:** Our entire modules are written in C++, and are designed to be readily used with handy commands as in §III. LT-SLAM’s pose-graph optimizer is implemented using iSAM2 [38] of GTSAM [39]. We adopted publicly available sources of Scan Context [10] and Removert [8]. We refer the readers our open source codes<sup>2</sup> for the specific parameters of the system. For the initial graph construction to be fed as an input of LT-mapper, we provide keyframe information saver<sup>3</sup> (i.e., pose-graph and global descriptors) as add-ons of existing LiDAR odometry open sources (e.g., LIO-SAM [26]).

**2) Datasets:** For the validation, Mu1Ran [4] and our own LT-ParkingLot dataset were selected. Both datasets have multiple sequences and repeated coverage on fixed sites.

**Mu1Ran dataset:** We leveraged this dataset to evaluate the feasibility of our LT-SLAM. Recently, we have acquired and released an extended sequence for KAIST that is suitable for long-term change detection research. We used the KAIST and DCC sequences to identify long-term changes. We note

that KAIST 04 was collected on Feb. 2021, which has an almost 1.5-year gap between KAIST 01 (Jun. 2019).

**LT-ParkingLot dataset:** A parking lot would be a typical place to witness LD changes. We collected six sessions at different times over three days. The sessions’ origins are all different and their global alignments are initially unknown as in the middle of Fig. 4(b).

### B. Multi-session Trajectory Alignment

Both qualitative and quantitative results for LT-SLAM are shown in Fig. 4 and Fig. 8. We used the RPG trajectory evaluation tool [40]. The intra-session translation and rotation (particularly yaw) errors are noticeably reduced via the inter-session anchor node-based loops. Two sessions with different origins successfully suppressed each other’s drifts.

### C. Lifelong Mapping

LT-mapper can update the world representation in two ways, as shown in Fig. 9. First, LT-mapper can efficiently maintain a *live map* via sending only LD changes to a central server, instead of whole snapshots. For the second representation, *meta map*, LT-mapper extends spatial volumes without adding weak PDs. This elaborates a meta representation of a 3D scene, which is independent of short-term stationary or periodic changes.

### D. Change Composition

Because there exist no point-wise ground-truth for the 3D changes over time, we propose an implicit way to

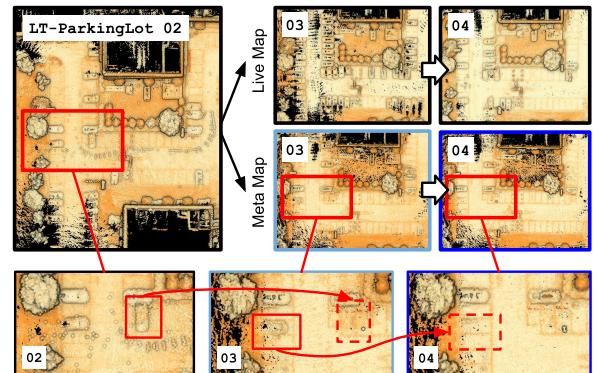


Fig. 9: LT-mapper supports two types of map management: *live map* and *meta map*. In the live map, the up-to-date representation of a scene is efficiently maintained. In the meta map, non-volume-maximizing points are iteratively removed (red boxes) while other persistent structures remain and be enhanced.

<sup>2</sup><https://github.com/gisbi-kim/lt-mapper>

<sup>3</sup><https://github.com/gisbi-kim/SC-LIO-SAM>

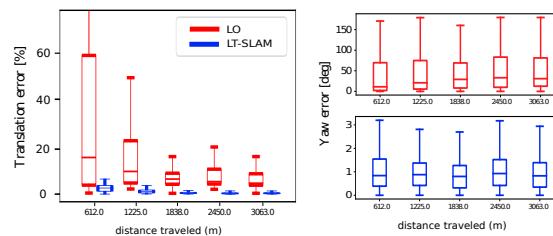


Fig. 8: The quantitative results of LT-SLAM on KAIST 01 and KAIST 02. The baseline LO is obtained on KAIST 01 by running LIO-SAM [26] without loop closures. We note that the error was resolved via MSS and inter-session loops despite no initial alignment of KAIST 02 being known.

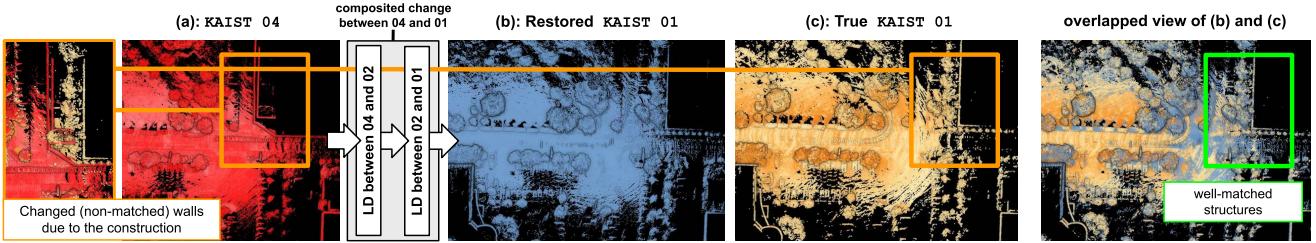


Fig. 10: An example of change composition. By chaining *delta maps*, LT-map can restore the map at any timestamp. After the rollback with change compositions, the unmatched wall in KAIST 04 (red) correctly disappears and the original wall in KAIST 01 is successfully recovered.

TABLE I: Accuracy evaluation of Fig. 10 using Chamfer distance and its statistics to summarize the structural inconsistency. Among the number of patches (NP) containing at least 25 points ( $NP_{valid}$ ), we count NP having the distance value larger than a threshold ( $\tau$ ).

Chamfer Distance (CD)	Max	Avg	Var	$\tau = 1$	$NP_{CD > \tau}$	$\tau = 2$	$\tau = 3$	$NP_{valid}$
Pos. Pair (01 $\leftrightarrow$ Restored 01)	6.41	0.29	0.31	38	9	1	1424	
Neg. Pair (01 $\leftrightarrow$ 04)	29.22	0.51	1.67	100	20	8	1386	

TABLE II: Efficiency evaluation of Fig. 10. LT-mapper efficiency against the snapshot-based method.

NOTE: for 100 KF <sub>01</sub> s and near 200 KF <sub>04</sub> s in Fig. 10	Memory Usage [MB] (merged map)	Computation Time [sec] (between 04 and 01)
Baseline (saving whole snapshot)	213.6	87.0 (w/o HD removal) 160.0 (w/ HD removal)
Ours (LT-map, delta map chaining)	85.7	9.8
Efficiency ratio	60% saved	8.9 (16.3) times faster

qualitatively evaluate our LD change detection performance via composing changes (Fig. 10). We have pre-calculated LD changes from LT-removert between KAIST 01 and 02, 02 and 04, except for 01 and 04. If the calculated  $LD_{01}^{02}$  and  $LD_{02}^{04}$  are reliable, then the composed virtual LD change  $\hat{LD}_{01}^{04}$  should match the actually obtained  $LD_{01}^{04}$ . In other words,  $\mathcal{P}_{04} \odot \hat{LD}_{04}^{01}$  should be equal to  $\mathcal{P}_{01}$ . As can be seen in Fig. 10, the restored KAIST 01 from 04 is well-matched to the real map of KAIST 01. This *delta map chaining* process is identical to map *rollback*, and we can restore a map at any timestamp without saving all memory-consuming snapshots.

**Accuracy:** To quantitatively evaluate the consistency between the restored 01 and 01, we use Chamfer distance [41]. First, we divide the aligned maps in Fig. 10 into  $5\text{ m}^3$  cubic patches and calculate the distance for each pair of corresponding patches having more than at least 25 points in a cubic. Table I shows the positive pair (i.e., restored 01 and 01) had lower distances and less inconsistent patches for a given target map 01 than the negative pair (e.g., 04 and 01) are reported.

**Efficiency:** For spatial change analysis, our change composition has also an advantage in computational efficiency. The time cost of conducting LT-removert once is  $\mathcal{O}(nm)$ , where  $n$  is the number of keyframes and  $m$  is the number of map points. Running LT-removert for a pair of consecutive sessions (i.e., between time  $t$  and  $t+1$ ) is required only once. Later when we aim to compare two arbitrary sessions, we need only to perform the lightweight change composition (empirically 0.05 sec per keyframe) which is linear to the

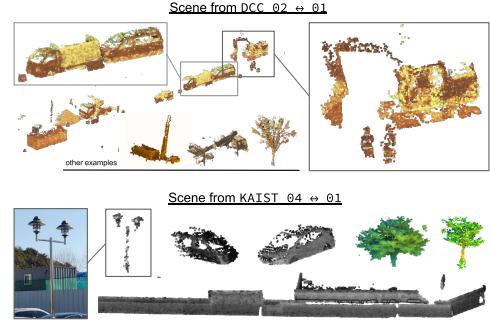


Fig. 11: Automatically parsed object examples. Top: the strong ND points in the scene of Fig. 6. Bottom: examples from the scene of Fig. 1. Colors were arbitrarily selected for clear visualization. All objects are parsed with simple Euclidean distance-based clustering.

*n*. Only from consecutive changes, we can efficiently make any combinatorial pair of sessions' changes without re-performing LT-removert for the pair. The quantitative report is summarized in Table II. Compared to maintaining entire sessions, our LT-map representation with *delta map* saved nearly 60 % of the amount of memory and yielded a performance 8.9 times faster than computing LD changes from scratch.

#### E. Automatic Parsing of Ephemeral Objects

From our PD and ND maps, we can easily segment ephemeral objects' points, as in Fig. 11. We expect this to promote understanding of the relationship between the ephemerality of an object and its 3D shapes. If we proactively assess the ephemerality of a 3D object, we also expect this to improve LT-removert performance via serving it as prior information.

## VI. CONCLUSION

In this work, we presented **LT-mapper**, which is an open, modular, and unified LiDAR-based lifelong mapping framework. We tackled three challenges to build a reliable long-term (day to year scale) map update system: 1) inconsistent ground-truths poses over sessions, 2) noisy points from high dynamic objects, and 3) new/disappeared structures. As shown in the results, our framework can be a core engine for multiple applications for urban map management and spatial understanding from efficiently maintaining live/meta maps, composing changes, to automatically labeling ephemeral shapes. In future work, we would integrate semantic information as well as the current geometric visibility-based approach for the weak ND and strong PD classification for more high-level change updates.

## REFERENCES

- [1] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, “Long-term 3d map maintenance in dynamic environments,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.* IEEE, 2014, pp. 3712–3719.
- [2] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, “Multiple relative pose graphs for robust cooperative mapping,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2010, pp. 3185–3192.
- [3] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt, “Metarooms: Building and maintaining long term spatial models in a dynamic world,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.* IEEE, 2014, pp. 1854–1861.
- [4] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim, “MulRan: Multimodal Range Dataset for Urban Place Recognition,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2020.
- [5] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph slam: Long-term mapping in low dynamic environments,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.* IEEE, 2012, pp. 1871–1878.
- [6] L. Wellhausen, R. Dubé, A. Gawel, R. Siegwart, and C. Cadena, “Reliable real-time change detection and mapping for 3d lidars,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, 2017, pp. 81–87.
- [7] J. Schauer and A. Nüchter, “The Peopleremover — Removing Dynamic Objects From 3-D Point Cloud Data by Traversing a Voxel Occupancy Grid,” *IEEE Robot. and Automat. Lett.*, vol. 3, no. 3, pp. 1679–1686, 2018.
- [8] G. Kim and A. Kim, “Remove, then Revert: Static Point cloud Map Construction using Multiresolution Range Images ,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, Las Vegas, Oct. 2020.
- [9] Y. Kim, J. Jeong, and A. Kim, “Stereo camera localization in 3d lidar maps,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 1–9.
- [10] G. Kim and A. Kim, “Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 4802–4809.
- [11] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, “Street-View Change Detection with Deconvolutional Networks,” in *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, June 2016.
- [12] M. Labb   and F. Michaud, “Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.
- [13] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, “maplab: An open framework for research in visual-inertial mapping and localization,” *IEEE Robot. and Automat. Lett.*, vol. 3, no. 3, pp. 1418–1425, 2018.
- [14] R. Elvira, J. D. Tard  s, and J. M. M. Montiel, “Orbslam-atlas: a robust and accurate multi-map system,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2019, pp. 6253–6259.
- [15] C. Campos, R. Elvira, J. J. G  mez, J. M. M. Montiel, and J. D. Tard  s, “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM,” *arXiv preprint arXiv:2007.11898*, 2020.
- [16] E. Palazzolo and C. Stachniss, “Fast image-based geometric change detection given a 3d model,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.* IEEE, 2018, pp. 6308–6315.
- [17] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, “Lifelong localization in changing environments,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.
- [18] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, “Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.
- [19] N. Banerjee, D. Lisin, J. Briggs, M. Llofriu, and M. E. Munich, “Lifelong Mapping using Adaptive Local Maps,” in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–8.
- [20] T. Kraj  k, J. P. Fentanes, J. M. Santos, and T. Duckett, “Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments,” *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.
- [21] W. Ding, S. Hou, H. Gao, G. Wan, and S. Song, “Lidar inertial odometry aided robust lidar localization system in changing city scenes,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.* IEEE, 2020, pp. 4322–4328.
- [22] H. Lim, S. Hwang, and H. Myung, “Erasor: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3d point cloud map building,” *IEEE Robot. and Automat. Lett.*, vol. 6, no. 2, pp. 2272–2279, 2021.
- [23] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 4758–4765.
- [24] Y. Cho, G. Kim, and A. Kim, “Unsupervised geometry-aware deep lidar odometry,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.* IEEE, 2020, pp. 2145–2152.
- [25] Z. Li and N. Wang, “DMLO: Deep Matching LiDAR Odometry,” 2020.
- [26] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2020.
- [27] M. Yokozuka, K. Koide, S. Oishi, and A. Banno, “Litamin: Lidar-based tracking and mapping by stabilized icp for geometry approximation with normal distributions,” 2020.
- [28] L. He, X. Wang, and H. Zhang, “M2DP: a novel 3D point cloud descriptor and its application in loop closure detection,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2016, pp. 231–237.
- [29] M. A. Uy and G. H. Lee, “PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition,” in *Proc. IEEE Conf. on Comput. Vision and Pattern Recog.*, 2018, pp. 4470–4479.
- [30] X. Chen, T. L  be, A. Milioto, T. R  hling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, “OverlapNet: Loop Closing for LiDAR-based SLAM,” in *Proc. Robot.: Science & Sys. Conf.*, 2019.
- [31] X. Xu, H. Yin, Z. Chen, Y. Wang, and R. Xiong, “DiSCO: Differentiable Scan Context with Orientation,” *arXiv preprint arXiv:2010.10949*, 2020.
- [32] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, “Real-time 6-dof multi-session visual slam over large-scale environments,” *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1144–1158, 2013.
- [33] P. Ozog, N. Carlevaris-Bianco, A. Kim, and R. M. Eustice, “Long-term mapping techniques for ship hull inspection and surveillance using an autonomous underwater vehicle,” *Journal of Field Robotics*, vol. 33, no. 3, pp. 265–289, 2016.
- [34] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.
- [35] G. Kim, S. Choi, and A. Kim, “Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments,” *IEEE Trans. Robot.*, pp. 1–19, 2021.
- [36] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust Map Optimization using Dynamic Co-

- variance Scaling,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.*, 2013, pp. 62–69.
- [37] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, “Pairwise consistent measurement set maximization for robust multi-robot map merging,” in *Proc. IEEE Intl. Conf. on Robot. and Automat.* IEEE, 2018, pp. 2916–2923.
- [38] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “iSAM2: Incremental smoothing and mapping using the Bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.
- [39] F. Dellaert, “Factor graphs and GTSAM: A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012.
- [40] Z. Zhang and D. Scaramuzza, “A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry,” in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018.
- [41] H. Fan, H. Su, and L. J. Guibas, “A point set generation network for 3d object reconstruction from a single image,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.