LILOC: Leveraging LiDARs for Accurate 3D Localization in Dynamic Indoor Environments

DARSHANA RATHNAYAKE, Singapore Management University, Singapore
MEERA RADHAKRISHNAN, University of Technology Sydney, Australia
INSEOK HWANG, POSTECH, South Korea
ARCHAN MISRA, Singapore Management University, Singapore

We present *LiLoc*, a system for precise 3D localization and tracking of mobile IoT devices (e.g., robots) in indoor environments using multi-perspective LiDAR sensing. *LiLoc* stands out with two key differentiators: Firstly, unlike traditional localization approaches, our method remains robust in dynamically changing environments, adeptly handling varying crowd levels and object layout changes. Secondly, *LiLoc* is independent of pre-built static maps, employing dynamically updated point clouds from infrastructural-mounted LiDARs and LiDARs on individual IoT devices. For fine-grained, near real-time tracking, *LiLoc* intermittently utilizes complex 3D "global" registration between point clouds for robust spot location estimates. It further complements this with simpler "local" registrations, continuously updating IoT device trajectories. We demonstrate that *LiLoc* can (a) support accurate location tracking with location and pose estimation error being <=7.4cm and <=3.2° respectively for 84% of the time and the median error increasing only marginally (8%), for correctly estimated trajectories, when the ambient environment is dynamic, (b) achieve a 36% reduction in median location estimation error compared to an approach that uses only quasi-static global point cloud, and (c) obtain spot location estimates with a latency of only 973 msecs. We also demonstrate how *LiLoc* efficiently integrates low-power inertial sensing, using a novel integration of inertial-based displacement to accelerate the local registration process, to enhance localization energy efficiency and latency.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; • **Computer systems organization** → *Sensors and actuators.*

Additional Key Words and Phrases: LiDAR, 3D Localization, Pose Estimation, Trajectory Tracking, Dynamic Indoor Environments

1 INTRODUCTION

Driven by a sharp drop in component prices, LiDAR sensors are being increasingly embedded to support 3-D sensing in a variety of IoT and mobile devices, such as cleaning robots [1], service robots [25] and smartphones/tablets [2]. Such high-resolution 3-D depth ranging supports capabilities such as obstacle avoidance, ambient object identification [30] and dimension estimation [28], and has been shown to pose new threats such as indirect eavesdropping on human speech [34]. In this paper, we investigate whether and how such LiDAR-generated 3D point clouds, can be integrated with conventional inertial sensing to develop a practical, lightweight, and accurate **localization** system for mobile IoT devices (such as robots) in *highly dynamic indoor environments*.

We believe that a LiDAR-based indoor localization solution offers another design point for indoor localization, with arguably some advantages over various various indoor localization technologies developed over the past two decades. Indeed, researchers have investigated a variety of sensing technologies for localization, such as wireless (including WiFi [19, 42], RFID [16], UWB [27]), acoustic and ultrasonic [22], inertial [17] and RGB vision [26], but low-cost, accurate indoor location solutions for high-occupancy, *dynamic* indoor spaces (such as university campus buildings

[1] https://us.roborock.com/pages/roborock-s5-max
[2] https://www.apple.com/sg/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/

Authors' addresses: Darshana Rathnayake, Singapore Management University, Singapore, darshanakg.2021@phdcs.smu.edu.sg; Meera Radhakrishnan, University of Technology Sydney, Australia, meeralakshmi.radhakrishnan@uts.edu.au; Inseok Hwang, POSTECH, South Korea, inseokh@postech.ac.kr; Archan Misra, Singapore Management University, Singapore, archanm@smu.edu.sg.

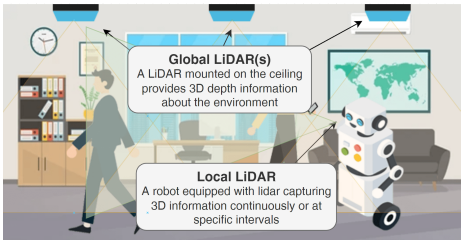


Fig. 1. *LiLoc* Overview: Using (GPC, LPC) Registration

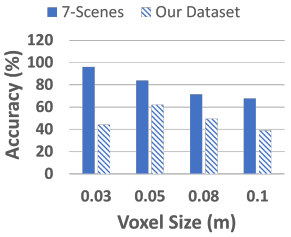


Fig. 2. Registration Accuracy for point clouds captured in same (7-Scenes) vs different (Our dataset) perspectives

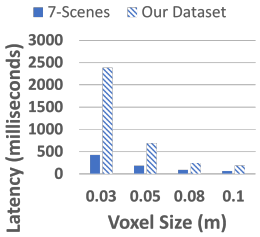


Fig. 3. Registration Latency vs. Point Cloud Voxel Size for the two datasets

and shopping malls) are still largely elusive. Broadly speaking, *fingerprinting* is a popular localization mechanism, where real-time sensor data generated by the IoT/mobile device is compared with offline data/features created during a prior training phase to determine the device's location–examples include RADAR [1] and Horus [45] for WiFi-based localization and RGB-SLAM [26] for vision-based localization. Creating such fingerprints is often a tedious, manual process and their performance is especially brittle in highly dynamic (e.g., [43]), where such fingerprints rapidly become obsolete.

We propose *LiLoc*, a new approach for LiDAR-based localization that is specially designed for dynamic environments. Note that LiDAR sensing, by itself, provides a device an ego-centric 3D map of its surroundings; while such a map is adequate for functions such as collision or obstacle avoidance, it cannot directly be used to generate a device's *global* location coordinates without additional secondary data. Figure 1 illustrates the central concept of *LiLoc*. In this approach, one or more *stationary*, ceiling/wall-mounted LiDAR sensors (with known location coordinates) form the infrastructure-based substrate. These static, infrastructurally-mounted sensors continuously scan the environment, generating a real-time, up-to-date *global* point-cloud (**GPC**), albeit from an elevated or "birds-eye" perspective. An individual mobile IoT device utilizes its own LiDAR sensor, typically located at or near floor level, to generate its own 3D *local* point-cloud (**LPC**), capturing a *partial* view of the environment from its egocentric perspective. *LiLoc* then employs 3D *registration* between these point clouds, ideally computing a spatial transformation that maps $LPC \rightarrow GPC$; this spatial transformation can then be used to localize the IoT device by transforming its unknown location (origin of the LPC) to the global coordinate system. To support efficient localization, this core function is invoked only intermittently (ideally, on-demand) and augmented with additional self-registration (called *local* registration) of successive LPCs generated by the IoT device. To overcome both (a) the relatively high power burden of LiDAR sensing and (b) the significant computational overhead of 3D point cloud processing, we also explore the interleaved use of inertial sensing-based dead reckoning to obtain an initial transformation matrix for the registration process. This approach allows *LiLoc* to significantly reduce the overall computational burden of point cloud manipulation, resulting in significant energy and latency savings.

While this mechanism of point cloud transformation via registration may superficially appear to be similar to visual or 3D SLAM techniques, developing a practical *LiLoc* implementation requires us to tackle several distinct, largely unappreciated, challenges:

(1) The GPC and LPC point clouds both represent the same (or a subset of the same) physical environment, but from very different perspectives: elevated ceiling-level vs. ground-level. Most past work on visual and 3D SLAM have

implicitly assumed that the reference and local point clouds have identical (or similar) perspectives. As illustrated in Figure 2 (and detailed in Section 3.1.5), state-of-the-art 3D registration algorithms often perform very poorly as the angular separation between the global and local perspectives increases. Here, we compare the registration accuracy for 7-Scenes dataset [3] (which has point clouds captured from the same ground-level perspective) vs. our dataset (with point clouds captured from very different (i.e., ceiling vs ground-level) perspectives).

(2) In dynamic, crowded environments, the LPC captured by the IoT device may suffer from various non-deterministic impairments. Due to its likely position at or near ground level, the LiDAR sensor's view may be *occluded*, with nearby objects (e.g., humans walking near a service robot) potentially blocking the LiDAR's ability to capture more distant objects. Alternately, due to the IoT object's movement, the LiDAR sensor may occasionally capture less-informative, *featureless* views (e.g., when pointing towards a wall) that imperil successful registration.

(3) The registration process itself involves a *precision-vs.-complexity* tradeoff that makes real-time localization (essential for wayfinding) challenging on IoT platforms. Precise centimeter-level localization often requires denser point clouds (fine-grained voxels), but this results in significant registration *latency*, even when offloaded to non-mobile devices. As shown in Figure 3 (and detailed in Section 5.2), a single (GPC, LPC) registration performed in a university lab room incurs ~2.23 secs of latency (for 3cm-sized voxels) on a laptop-grade device.

We shall show how a practical *LiLoc* system can overcome these challenges. In particular, we utilize two different feature extractors to balance accuracy vs. cost: (a) FCGF [5] to extract global features (from the GPC) and (b) FPFH [33] to extract local features (from the LPC). To improve the performance of *LiLoc*-based location tracking, we introduce (a) a grid-based search method to reduce the total computational complexity of (GPC, LPC) registration and (b) develop a set of outlier/noise filtering strategies to *predictively* identify and eliminate GPC pairs that are likely to suffer from poor registration accuracy. To further reduce the latency incurred by multiple executions of RANSAC [9] (during grid-search), we propose to utilize random sampling of points (with their features), an approach that is computationally cheaper but preserves registration accuracy. We implement the *LiLoc* system on a commodity laptop and show that this combination of techniques can help support **fine-grained, near real-time** location tracking (median location and pose estimation errors <= 7.4cm and <=3.2°, respectively) across multiple real-world dynamic environments.

Key Contributions: We make the following key contributions:

- *Precise 3D localization using Multi-perspective 3D point clouds*: We introduce a novel approach that utilizes a deep learning model (applied to a continually updated GPC, thereby avoiding the problem of stale fingerprints), 3D vision primitives and system level optimizations for robust LiDAR-based 6D (location, angle) pose estimation of moving objects in a dynamic environment. It uses a grid-based RANSAC registration method and a proactive elimination of less-informative point cloud instances to enhance the accuracy (by ~12.5%) together with judicious voxel downsampling and random sampling of points to improve the latency by ~56% (i.e., a reduction from 2.23 secs to 0.97 secs).
- *Fuse Global and Local Registration for Efficient Continuous Localization*: We develop a localization strategy, where the complex *global* registration (GPC-LPC registration) is performed only intermittently and on-demand to obtain robust 'spot' estimates of an IoT device's location/pose. In the intermediate time period, the IoT device performs repeated simpler *local* registrations, across closely-spaced LPC frames sensed as the IoT device moves around, to update the device's trajectory. As such local registration can occasionally fail due to deficiencies in the captured point cloud, we develop a variance-based metric that can proactively identify and eliminate such deficient point clouds, thereby

[3] https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes/

improving the success rate of local registration by a further ~15%. In addition, we show how to interweave such LiDAR-based localization with lower-power inertial sensing-based dead reckoning: the combined and optimized system enables a 50+% decrease in the energy overhead of continuous localization.

- *Demonstrate the Significant Performance Gains of* LiLoc: We implemented multiple variants of a functional *LiLoc* system, by using 3 global static LiDARs and a moving LiDAR (carried by the user). Experimental studies at a multi-occupant university indoor space (with an area equal to $80m^2$) demonstrate that: (a) *LiLoc* is accurate, supporting continuous location tracking, of objects with human-like motion, with location estimation error being <=7.4cm 84% of the time, and (b) *LiLoc* is robust to environmental dynamics, with the median location/pose estimation error increasing only modestly from 0.178 m/11.35° to 0.216 m/16.57° when the number of moving humans in the ambient environment increases from 1 to 5. We further demonstrate the importance of continuous GPC sensing: using a quasi-static global point cloud, collected even once a day, results in a significant 36% increase in median location estimation error compared to *LiLoc*. We also demonstrate that how the use of octree-based point cloud compression techniques can reduce the network latency of transferring point clouds significantly, by over 46%.

2 RELATED WORKS

We discuss prior works on LiDAR-based localization and summarize localization approaches using other technologies.
LiDAR-SLAM: There have been existing works on LiDAR-based simultaneous localization and mapping (SLAM), which primarily utilizes scan-to-scan matching [4, 32] or feature-based matching [10] to estimate the location. These approaches are primarily developed for expensive mechanical LiDAR sensors which collect the surrounding information by spinning a high frequency laser array. In our work, we utilize the solid state LiDAR devices, which are built entirely on a silicon chip with no moving parts involved. Similar to SLAM approaches, we leverage the Iterative Closest Point (ICP) algorithm [4] in our approach to estimate and refine the final pose. The LOL approach [32] uses LiDAR-only odometry for identifying similar locations between an online 3D point cloud and an apriori offline map. Koide et al. [18] propose a portable LiDAR-based system for monitoring the behavior of people. This approach first uses graph SLAM for offline environment mapping and further leverage angular velocity and range data provided by the LiDAR for estimating sensor pose. These approaches require high computational resources and have low update frequency. V-LOAM [47] integrates a monocular camera and a mechanical 3D LiDAR to estimate motion as well as build a map of the traversed environment. One of the main open issue with LOAM pipelines is the odometry drift due to the accumulated errors. The HIDA system [23] utilizes a solid state LiDAR point cloud for instance segmentation and further to aid in indoor navigation for the visually impaired. RTAB-Map (Real-Time Appearance-Based Mapping) [20] is an open source SLAM system that utilizes both visual and geometric information for mapping and localization. It identifies previously visited locations and improve map consistency over time. Another SLAM system, Google Cartographer [11], is designed to work with 2D and 3D data from LiDAR to create accurate maps of indoor and outdoor environments. It uses scan matching and loop closure techniques to optimize the robot's trajectory and map consistency. In a more recent work [37], the authors propose a new static SLAM framework based on a solid-state LiDAR for high frequency localization. Unlike these works, our approach works on a dynamic global map captured from a very different perspective and is suitable for localizing in dynamic environments. These existing SLAM techniques can be adapted for local registration purposes. In this work, we adopt a straightforward yet efficient approach, which involves registering consecutive point clouds and utilizing a technique to assess the informativeness of the point clouds for trajectory segmentation.
Navigation of Autonomous Vehicles: A recent survey article [8] on the use of 3D LiDARs for navigation of autonomous vehicles report that approaches such as 3D registration methods, feature detecting & matching and deep

learning models are leveraged for accurately localizing the vehicle. In 3D registration methods ([36], [49]), either the incoming scans with portions of pre-built point cloud map is used to localize the vehicle or successive LiDAR scans are combined to calculate the odometry of the vehicle. While deep learning methods ([24], [44]) are producing promising results in odometry and location estimation, 3D feature-based methods ([15], [14]) are more robust in real life scenarios and are still considered the state of the art. The survey also reports that the proposed techniques suffer from the presence of dynamic objects (e.g., pedestrians, cars) in the environment when trying to regress the motion between two frames and such dynamic objects are removed from the scene to obtain more accurate odometry results.

Indoor Localization Systems: Traditional localization approaches that leverage on external setup, are affected by environmental uncertainties and lack robustness in different dynamic scenarios. For example, (a) WiFi-based indoor positioning technologies require mechanisms such as triangulation [41], accurate fingerprinting of the environment and multiple routers (to achieve high accuracy) [19, 42, 50], inter/extrapolation [35], or crowdsourcing-based technologies [40]. However, these approaches have its own limitations (based on a survey by Jang et al. [13]) especially in certain indoor environments (e.g., harsh radio environments in factories/warehouses, environments such as malls or performance arenas where the spatial layout changes varies frequently), where a *LiLoc*-like localization approach could prove to be more robust, (b) UWB localization [7, 27] requires pre-installation and calibration of multiple anchors. While UWB positioning can attain centimeter-level accuracy in three-dimensional space, achieving 6 DOF pose estimation (necessary for VR and adavanced robotics applications) remains an ongoing challenge [48]. Despite operating over longer coverage distances, UWB systems may experience accuracy issues due to factors such as signal interference, multipath propagation and human body-driven attenuation [38], especially in dense and dynamic indoor environments, (c) visible light-based localization [21, 46] can work on existing infrastructure but requires complex programming and controlling of the individual bulbs in a wireless manner, (d) RF-based approaches (e.g., Zigbee [12], BLE [31], RFID [16]) are affected by environmental artefacts (i.e., weather change, dynamicity of the environment), and (e) pure vision-based systems [26] have significant privacy concerns, are affected by illumination change and often have poor accuracy in multi-occupant settings. The recent advancements in solid state 3D LiDAR devices shows significant promise in overcoming several of the challenges mentioned above.

3 PRELIMINARIES

We envision a low-cost, privacy-preserving, and robust system for precise localization of humans and moving objects in a *dynamic indoor environment* (e.g., shopping mall, hospital). For instance, imagine a shopping mall where there are multiple moving humans and smart objects (e.g., guidance robots that provide navigation assistance or other delivery robots that drop off packages).

The proposed LiDAR-based *LiLoc* approach would allow a guidance robot to lead shoppers to specific stores or products while navigating around obstacles and crowds. *LiLoc* can also be used for augmented reality (AR) applications in dynamic indoor environments, such as museums and exhibitions, to provide accurate positioning and orientation information to the user's device. This can enable immersive experiences where virtual objects and information content are seamlessly overlaid on the physical environment. Given these, we propose a scenario where the indoor environment is instrumented with stationary LiDAR devices mounted at known location coordinates (which we denote as the "*global LiDAR*"), while the humans (e.g., mounted on a smartglass) or IoT devices are equipped with a LiDAR sensor (which we call the "*local LiDAR*"). Note that multiple stationary LiDARs are needed only to ensure adequate coverage of the area of interest, and is a function of an individual device's range and field of view. As an alternative, an individual spinning LIDAR, which provides a 360° field of view (FoV), may also be used to provide the global FoV. The global

Table 1. Specifications of the LiDAR device

Sensor	Frequency	FoV	Horizontal Resolution	Vertical Resolution	Detection Range	Accuracy
RealSense L515	30 Hz	70° x 55°	0.07°	0.07°	0.25-9m	1.4 cm
iPad Pro LiDAR	–	120° x 30°	0.2°	0.2°	~5m	1.0 cm
Mechanical LiDAR	5-20Hz	360° x 32°	0.1-0.4°	2.0°	0.5-100 m	3.0cm

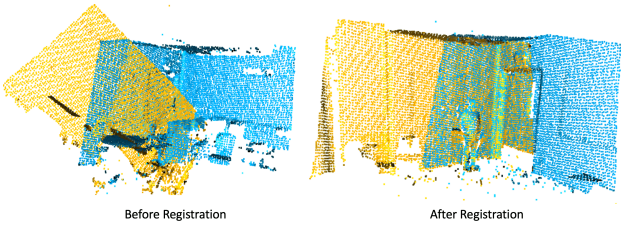


Fig. 4. Example of 3D Point Cloud Registration

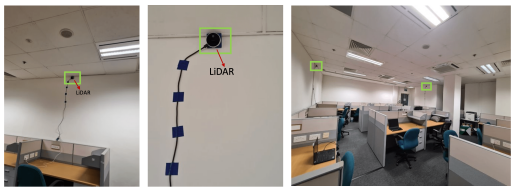


Fig. 5. LiDAR placement in the indoor space

LiDAR scans continuously to generate an up-to-date 3D rendering of the environment, whereas secondary LiDAR captures egocentric point clouds intermittently. At a very high level, our proposed system should take as input a global point cloud (GPC) and a secondary point cloud (SPC), determine the correct transformation to align the two point clouds correctly and thereby estimate the location and/or pose of the secondary LiDAR source (i.e., the pose of the person or robot).

3.1 LiDAR for Localization: Empirical Findings

Before describing our solution, we present some empirical insights into available LiDAR technologies and their implications for *LiLoc's* design.

3.1.1 Choice of LiDAR:. Compared to a traditional mechanical LiDAR, a solid state LiDAR is cheaper, and has a shorter range, higher angular resolution, and higher update frequency. As the angular resolution is higher, the points are more dense within the same scanning area. However, a main limitation of solid-state LiDARs is their limited field of view. These LiDAR sensors are now available as standalone commercial devices (e.g., Intel RealSense L515 [4]), as Time of Flight camera modules for Raspberry Pi devices [5], on COTS smartphones (iPhone models 12 and above, Samsung Galaxy S20 Plus and Ultra) and will be integrated into even smartglasses (such as the upcoming Apple VisionPro™). We thus envision that such inexpensive LiDAR sensors would also be embedded in robotic units and wearables, and can be automatically triggered without human intervention. Table 1 shows the comparison of the specifications of Intel RealSense LiDAR, iPad Pro LiDAR, and mechanical LiDAR. In this work, we leverage the Intel RealSense LiDAR which is compact (61mm x 26 mm), lightweight (~95g), and easily programmable. Although the official specifications of this LiDAR mention a maximum range of 9 meters, in practice, we are able to obtain accurate depth estimates only up to 6 meters (as the range also depends on the environment's reflectivity).

3.1.2 Point cloud registration: As a background to our LiDAR-based localization approach, we first briefly describe the point cloud registration process, which is key to our approach. Point cloud registration is the process of finding the

[4]https://www.intelrealsense.com/lidar-camera-l515/
[5]https://www.arducam.com/time-of-flight-camera-raspberry-pi/

transformation matrix for a given source point cloud with respect to a target point cloud; i.e., the goal is to spatially align two point clouds by finding the correct translation, rotation, and scale. In our case, the scale remains the same for both GPC and LPC, therefore, the transformation matrix only contains the rotation and translation. Figure 4 shows an example of two point clouds before and after the registration process.

Point cloud registration can be categorized as either *coarse* or *fine*. Fine registration requires an initial transformation matrix which is generally estimated by a coarse registration algorithm. In this work, we combine both approaches to obtain an accurate registration. We utilize the Iterative Closest Point (ICP) algorithm [3] for fine registration and follows two different approaches to obtain the initial alignment for ICP registration. First, we utilize the RaNdom SAmple Consensus (RANSAC) algorithm [9] to perform pairwise registration of point clouds, given the two corresponding feature descriptors. However, this pure LiDAR-based registration approach imposes a higher power and computation overhead. Therefore, as an optimized alternative, we propose to leverage an inertial measurement unit (IMU) to obtain the initial estimate/hint for the point cloud transformation matrix. The Intel RealSense LiDAR is equipped with an IMU that has an accelerometer which measures acceleration values in meters per second squared (m/s^2) and a gyroscope which measures angular velocity in units of radians per second (rad/s). To minimize the errors, we first calibrate the IMU sensor. We use IMU measurements to estimate the relative motion between two point cloud captures, which is then used as an initial alignment for the ICP registration process. The IMU estimates can be interleaved with point cloud-based estimates to reduce the significant overhead that results from a LiDAR-only continuous 3D point cloud processing approach. Interestingly, we shall see that using the IMU as an enabler for coarse registration is superior to an alternative of using the IMU directly for dead reckoning-based motion tracking.

As an initial exploration, we utilize the 7-Scenes dataset which contains a collection of both RGB and depth images (captured using Kinect RGB-D camera), together with ground-truth camera-to-world transformation matrices. To develop solutions that accommodate the anticipated perspective divergence between our global and local LiDARs, we also curated our own dataset (described later in Section 7) with multiple Intel LiDARs. In our dataset, *pose* is the transformation matrix that aligns the local/secondary LiDAR cameras to the global cameras.

We define point cloud registration *accuracy* in terms of the difference (for both rotation and translation values) between the ground truth and estimated transformation matrices. For global registration (explained in detail, in Section 5.2), we label registration as accurate if the rotational error $\leq 10^\circ$ and the translational error $\leq$0.3m. For local registration (see Section 5.1), we set a tighter error upper bound of 3° for rotation and 0.1m for translation.

3.1.3 Point cloud network transfer. In our system, the LPCs are transmitted wirelessly over a WiFi network while the GPCs are transmitted over Ethernet. The wireless transmission of LPCs incurs significant network latency. On average, a raw point cloud contains 142K points, resulting in an empirical network transmission latency of ~283 msecs. Moreover, this time would increase significantly with simultaneous transmissions–e.g., when multiple moving robots are concurrently attempting to perform global registration. Voxelization offers one approach to point cloud size reduction: the number of points reduces to 12K and 5K for 0.03 and 0.05 voxel sizes, with a corresponding reduction in transmission time to 69 and 26 msecs, respectively. However, the loss of fine-grained features during voxelization can increase the pose estimation error. To reduce latency further with minimum loss of accuracy, we additionally explore the use of point cloud compression techniques. We study and employ Octree-based point cloud compression, whereby an octree is first constructed from the point clouds by partitioning the 3D space into 8 cubes of the same size. This lossy compression technique then recursively partitions each non-empty cube in the same way until the maximum depth level is reached. Eventually, a point is represented by the center of the corresponding cube. We empirically set

Table 2. Point cloud registration performance for global LiDAR placed at human level and ceiling level

	Accuracy	Coverage	No. of points
Human-level (1.5m)	60.21%	25.99%	51849
Ceiling-level (3.5m)	69.38%	34.74%	69304

Table 3. Overlap of point clouds for different angles of tilt of global LiDAR

Angle	15°	30°	45°	60°	75°
Overlap ratio	62.5%	50.4%	34.7%	22.5%	16.5%

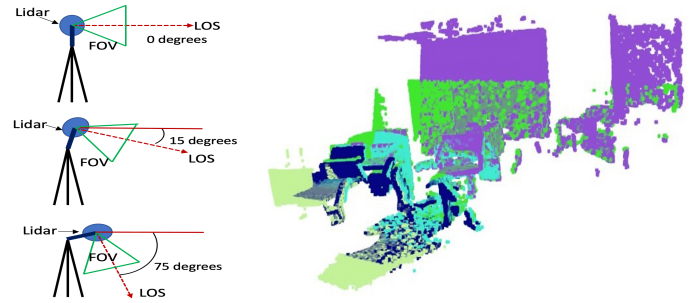
the octree depth to 8 as this offers optimal performance, achieving compression ratios ranging from 7x to 41x while suffering a reconstruction RMSE of only 0.03m.

3.1.4 Placement of the Global LiDAR:. The placement of the global LiDAR should offer a relatively unobstructed view (i.e., close to or at ceiling height), as well as provide sufficient coverage of the environment. Purely because of the limited field of view (70° x 50°) of current versions of RealSense LiDARs, multiple such devices are needed to effectively monitor a moderately large lab space. Figure 5 shows the placement of the LiDARs in our experimental indoor space. To enable judicious placement of the global LiDARs, we empirically analyzed the overlap ratio and the global registration performance under varying placement positions.

(1) **Registration results for varying mounting heights**: In the indoor space shown in Figure 5, we capture point cloud data by mounting the global LiDAR at two different heights: (a) human-level at about 1.5m (i.e., the same level as local/secondary device), and (b) ceiling-level at about 3.5m. To study the registration performance for varying positions, a reference point cloud was first captured by scanning the environment using an iPad Pro device. The captured reference point cloud (with voxel size=0.03m) has 199,444 points. Table 2 shows the point cloud registration accuracy, coverage with respect to the reference point cloud, and the number of points captured for the two different positions. We observe that the ceiling-level placement achieves about 9% higher coverage and registration accuracy compared to the lower human-level placement.

(2) **Overlap for different angles**: To study the coverage that can be obtained for varying angles of tilt of the LiDAR, we mounted the LiDAR on a tripod and experimented with tilt angle values of {15°, 30°, 45°, 60°, 75°}. The angle is measured as the 'pitch' (see Figure 6(a)), or the difference between the line of sight of the LiDAR camera and the floor. Table 3 shows the overlap of point clouds for different angles of tilt of the global LiDAR. Here, the overlap is calculated as a ratio of the field of view (FoV) captured at each tilt to the total FoV (i.e., the point cloud stitched from views of all angles). We observe that when the LiDAR is almost vertical to the floor (i.e., 75°), the overlap ratio is very low. Although the overlap is higher at 15°, the captured point cloud has a lot of 'featureless' points corresponding to walls. Accordingly, we empirically set the pitch angle to 30°, thereby ensuring reasonable coverage and capturing most of the geometric structure of the environment.

(3) **Number of LiDARs vs Area covered**: In this experiment, we mounted the LiDARs at 8 different positions within the indoor space. The point clouds from all positions are stitched together to obtain the reference point cloud. We then compute the overlap of the point clouds stitched from multiple views to the reference point cloud (i.e., $overlap = \frac{N_s}{N_r}$, where N_s is the number of points in a stitched point cloud and N_r is the number of points in the reference point cloud). Figure 7 shows the boxplot showing the minimum, maximum, and mean (indicated by orange line) coverage. As anticipated, higher coverage can be obtained by using a larger number of global LiDAR sensors.



(a) Lidar kept at different angles (b) Point clouds captured at different angles

Fig. 6. (a) Illustration of LiDAR placement at (0°, 15°, 75°) angles, (b) Point clouds for different angles (15°, 30°, 45°, 60°, and 75°) depicted in different colors

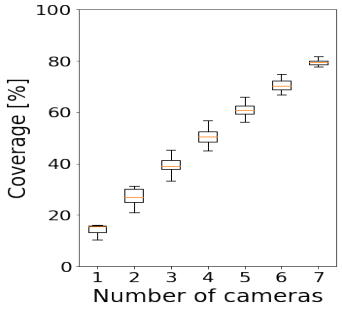


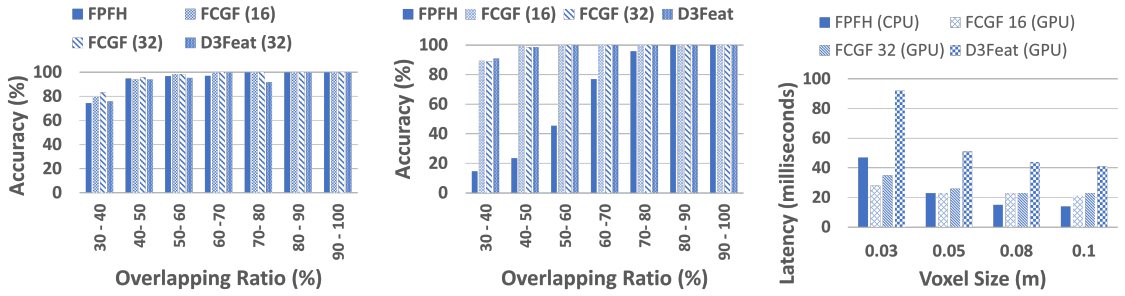
Fig. 7. The overlap for point clouds captured from multiple LIDARs kept at different positions.

Key Takeaway: We observe that 3 LiDAR devices mounted at $\sim 3.5m$ height and a tilt of 30° w.r.t wall are needed to achieve sufficient coverage (at least 40%) in an area of size 80 sqm.

3.1.5 Feature extraction on 3D point clouds: Traditional approaches for extracting feature detectors and descriptors from point clouds perform poorly in cluttered and dynamic environments. Recently, several deep learning-based detectors and descriptors have been proposed for 3D point clouds. We explore two of the recent state-of-the-art deep learning-based 3D point cloud feature extraction techniques: (a) D3Feat [2], and (b) FCGF [5], as well as a traditional alternative (c) FPFH [33] that has significantly lower computational overhead. We utilize 14 trajectories from the 7-Scenes dataset and 10 trajectories from our own dataset to evaluate the relative performance of these three feature extraction algorithms. It should be noted that the point clouds in 7-Scenes dataset have a smaller FoV and very high overlap ratio (i.e., the difference across subsequent frames is very low). In contrast, our dataset (which better mimics the *LiLoc* deployment scenarios) has a lower overlap ratio. For each trajectory, the overlap percentage is calculated for multiple combinations of point cloud pairs. The pairs that had less than 30% overlap between them are discarded. To perform a fairer comparison of registration efficacy, we dropped some intermediate frames in the 7-Scenes dataset to ensure that consecutive frames chosen for registration are not very-highly overlapping.

Figure 8 shows the comparison of point cloud registration accuracy and latency for the different feature extractors (FCGF (16 & 32), D3Feat (32), and FPFH (33) dimensions) on both datasets. We observe that D3Feat and FCGF achieve comparable registration accuracy (i.e., achieve >90% accuracy even when the overlap is only 40%) on our dataset. However, FCGF is 2.5 times faster than D3Feat, making it an ideal choice for the global registration process. While FPFH is less accurate on our dataset, it is more lightweight (runs on a CPU) and faster, and offers higher accuracy when the overlap ratio is higher. So, FPFH is ideal for performing repeated feature extraction (at lower latency) during local registrations, where successive frames can be anticipated to have higher overlap.

We study the registration accuracy and latency for different voxel sizes, Based on our feasibility studies (as shown earlier in Figure 2), we achieve a higher registration accuracy for voxel size=0.05 and comparable accuracies for voxel sizes 0.03 and 0.08 on our dataset. Moreover, the DNN-based FCGF feature extractor we utilize for global registration is based on a ResUNetBN2C architecture and is trained for a 0.025 voxel size. For the system design and performance evaluation, we opt for voxel sizes, 0.03 and 0.05, as they are within a reasonable range relative to the trained voxel size of the original model.



(a) Registration Accuracy (7-Scenes) (b) Registration Accuracy (our Dataset) (c) Registration Latency (7-Scenes)

Fig. 8. (a) Registration accuracy for 7-Scenes dataset for different overlap ratios and voxel size=0.03, (b) Registration accuracy for our dataset for varying overlap ratios and voxel size=0.03, (c) Latency of registration for different voxel sizes compared for different feature extractors.

Key Takeaway: We extract features based on a 16-dimension FCGF network on GPC for global registration. D3Feat and FCGF exhibit comparable registration accuracy (>90% even at 40% overlap), with FCGF being 2.5 times faster. This makes FCGF preferable for global registration, especially in a pervasive environment where the edge device may need to concurrently support multiple mobile devices. Conversely, FPFH, although less accurate, offers lightweight processing and higher accuracy at greater overlap ratios. Thus, FPFH is suited for obtaining robust multi-dimensional features from LPC during the more frequently invoked local registration process (where the overlap between consecutively sampled point clouds is fairly high). Based on initial findings, we utilize voxel sizes of 0.03 and 0.05 for the system design and evaluation.

4 SYSTEM OVERVIEW

In this Section, we outline the design goals, the challenges to realize *LiLoc* and the high-level architecture of our system.

4.1 Design Goals & Challenges

Low-cost & Privacy preserving approach: For greater adoption of the system, we require an approach that is low-cost and has minimal “visual" privacy concerns. We propose to instrument the environment with cheaper solid-state LiDAR devices that capture depth images/point clouds compared to expensive mechanical LiDARs. The information in a depth image is not sufficient enough to identify a person, especially in GPCs. In addition, our localization technique does not require storage of previously captured GPCs and performs all local registrations on the mobile ‘secondary’ device.

Precise Localization: We aim to provide centimeter-level localization accuracy in near real-time fashion. We focus on optimizing the existing, well-established point cloud registration algorithms on point clouds captured from two distinct perspectives.

Work in Dynamic Settings: *LiLoc* should work accurately in dynamic real-world conditions, e.g., passers-by of varying numbers, positions, and mobility resulting in dynamic scene changes and variable levels of occlusion in the point clouds.

To realize the above properties, our system needs to overcome several **key challenges:**

Finding & validating accurate registrations between 3D point clouds: Establishing accurate registrations between two pairs of point clouds is key to our system. However, this is challenging due to: (a) point clouds captured from two varied perspectives (e.g., global LiDAR on the ceiling and secondary LiDAR at human level) of the same scene having

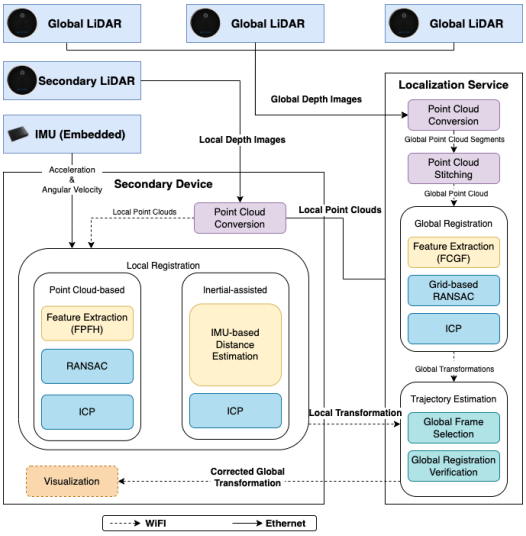


Fig. 9. *LiLoc's* High-level System Architecture

minimal overlap, (b) possibly varying resolutions of point clouds. It is also challenging to verify the correctness of the registrations.

Handle dynamic changes in the environment: Scene changes would demand frequent updates of the global point cloud and re-registration (i.e., $LPC \rightarrow GPC$ registration). This is computationally heavy and would affect the system latency. Therefore, it is important to devise a mechanism where lighter-weight computations are performed more frequently to update the device's trajectory.

Lack of structural diversity in the environment: Some environments possess very few structural features that might help with registration. This issue would be particularly aggravated for the secondary LPC captured by the mobile/IoT device, whose close-to-floor LiDAR positioning and unpredictable mobility might generate more frequent occlusion or views of feature-less scenes (e.g., plain walls).

Dealing with issues of 3D point cloud data: Leveraging 3D point clouds brings up additional challenges like (a) missing depth information or incomplete 3D scan data due to occlusion and partial surfaces, (b) sensor being noisy, limited FoV and range, (c) dynamic depth information causing changes in point cloud density (e.g., near vs. far objects).

4.2 System Architecture

In Figure 9, we illustrate *LiLoc's* high-level system architecture for localizing IoT devices and/or humans in a dynamic indoor environment. The architecture involves an indoor space statically instrumented with solid-state LiDAR devices and the IoT device/human (to be localized) equipped with a LiDAR sensor of their own.

The global LiDAR (mounted on the ceiling) captures the depth images of the indoor environment and transmits them to *LiLoc's Localization Service*. This service will be deployed within the infrastructure, accessible to secondary devices via WiFi connectivity. The secondary device, such as robots or smartphones equipped with a LiDAR (or depth) sensor, is assumed to possess some computational capability to execute specific components of the system. This distributed approach alleviates the computational load on the infrastructural-server and/or edge device and also reduces

the communication bandwidth needed for *LiLoc's* operation for multiple mobile devices. This approach additionally enhances user privacy, as not all point clouds captured by the secondary devices need to be transferred to the central service. The captured depth image is then converted to a point cloud using the known intrinsic parameters of the LiDAR. We obtain point clouds from multiple LiDARs mounted at known locations in the environment (as needed to capture the space of interest) and then stitch them together to obtain one *GPC*. A similar depth-to-point-cloud conversion is executed at the local secondary device. *LiLoc's* localization strategy involves two steps: (a) intermittent spot localization, and (b) continuous trajectory estimation.

Spot localization provides an instantaneous estimate of an IoT device or a person's location/pose. For this, the *LPC* at that instant is sent to the localization service and registered with the corresponding GPC. This process is termed as *Global Registration* (explained in detail later in Section 5.2). Trajectory estimation is an extension to spot localization that localizes a device/user continuously. In *LiLoc*, estimating trajectory is a three-step process. First, the LPCs are registered in the user's local space which is not currently aligned to any GPC. This is called *Local Registration*, which can be achieved in two different ways, as explained in Section 5.1. One approach relies solely on point clouds to compute the transformation matrices between consecutive LPCs whereas the alternative approach combines the LiDAR-based localization with a lower-power inertial sensing-based dead reckoning approach. The IMU-based approach, contingent upon the presence of an inertial sensor in the secondary device, utilizes the acceleration and angular velocity from the inertial sensor to estimate the transformation matrices. Additionally, it is presumed that the transformation matrix required to align the axes of the LiDAR sensor with those of the inertial sensor is provided. Notably, the IMU-based approach does not perform dead reckoning-based trajectory tracking, the approach commonly used in most IMU-supported prior localization work, as its accuracy is seen to be quite inferior to our proposed approach. Next, the secondary device sends LPCs and locally estimated transformation matrices to the *localization service* for global registration. It then takes account of multiple instances of LPCs and GPCs to verify the correctness of the global registration process. Once the verification process is completed, the global transformation matrices are relayed to the secondary device. The secondary device can then continue to convert local registrations to the global coordinate system.

5 LIDAR-BASED LOCALIZATION & TRAJECTORY ESTIMATION

We now detail the key components of *LiLoc*. Broadly speaking, we extract specific features on GPC & LPC and perform the complex global registration (GPC-LPC registration) intermittently to obtain snapshots of the IoT object's location. Such registration is interleaved with a sequence of simpler local (LPC-LPC) registrations, performed on the mobile device, which help to update the evolution in the device's trajectory between such snapshots.

5.1 Local Registration

It is computationally inefficient, bandwidth intensive and impractical to obtain the registration across a GPC and each frame of an LPC (which is captured more frequently). As such, we first compute the transformation between the subsequent local point cloud frames to estimate the trajectory in the user's local space (i.e., to capture relative motion or displacement between the local frames). We refer to this process as *local registration*. As introduced earlier in Section 3, we propose two approaches to perform the local registration depending on the availability of the IMU.

5.1.1 Point Cloud-Based Registration. This approach only makes use of point clouds to compute the relative motion of the device. Here, we exploit the fact that the overlap across subsequent point clouds captured, from the same perspective, by the secondary device would be high, and thus local registrations can be performed more efficiently. We extract *local*

features on LPC using Fast Point Feature Histograms (FPFH) technique. As shown earlier in Section 3.1.5, FPFH is fast and achieves high registration accuracy when the overlap across subsequent point clouds is greater than 60%. We then employ the RANSAC technique to perform the local registrations and further utilize ICP algorithm to refine the local transformations obtained.

5.1.2 Inertial-Assisted Point Cloud Registration. Inertial sensors, if available, can aid in estimating the initial transformation matrix for the ICP registration process, which bypasses the need for feature descriptor computation and RANSAC feature matching in the local registration process. This approach makes the process more lightweight and faster. The initial transformation matrix provides a rough alignment between two consecutive point clouds, which is further refined by the ICP algorithm. We sample the accelerometer and gyroscope sensor data from the Intel RealSense device at 200 Hz. To obtain accurate displacement information, we first eliminate the gravity component via a multi-stage process. First, we utilize a calibration step, lasting 4-5 secs while the device is stationary, to compute the gravity component readings along all 3 axes of the accelerometer's readings. As a first-level smoothing filter, we utilize a Gaussian filter with a standard deviation of two. Subsequently, when the device is moving, its new orientation pose is calculated by integrating the angular velocity. The gravity components measured during calibration are then rotated to the device's current coordinate system and subtracted from the accelerometer readings. Finally, a moving average filter is then applied to further smooth the signal.

As the sampling rate of the IMU sensor (~200 Hz) is higher than the LiDAR sensor (30 Hz), there will be several IMU readings between two consecutive point cloud captures. Starting from the closest reading to one point cloud, the distance is continuously estimated until the next point cloud is captured. Due to the high sampling rate of the IMU, the device is assumed to move under constant acceleration between two consecutive IMU measurements.

The steps mentioned above are depicted in Algorithm 1. The algorithm takes the gravity component-removed accelerometer's readings ($a[\,][\,]$), the angular velocity ($\omega[\,][\,]$) from the gyroscope's readings, and corresponding timestamps ($timestamps[\,]$), between the two consecutive point cloud captures. The total displacement, denoted by the $translation$ vector, and rotation, represented by the $rotation\_matrix$, are acquired by integrating a and ω over time. The relative distance (Δd) and orientation of the device (Δr) between consecutive inertial readings are aggregated to derive them. At each time step, the relative distance (Δd) is scaled by the rotation matrix ($rotation\_matrix$), which corresponds to the previous orientation of the device (Δr) relative to the initial inertial measurement ($t = 1$). The rotation matrix representing the device's current orientation is computed by sequentially multiplying the rotation matrices derived from the relative rotation (Δr) at each time step. The resulting transformation matrix combines the last translation vector and rotation matrix, which is then utilized in the ICP algorithm for more precise point cloud-based computation of the device's transformation matrix. The refined transformation matrix is employed to calculate the device's $velocity$ which serves as the initial velocity (as an argument to $CALCULATE\_INITIAL\_TRANSFORMATION$ function) to estimate the subsequent transformation matrix for the ICP-based registration. This process is illustrated in Figure 10.

5.2 Global Registration

Global registration finds the transformation across GPC and LPC (available at the same time instance) and aligns them. Here, we utilize the FCGF network [5] which jointly learns keypoints and descriptors from 3D point clouds to extract features on GPC. Using simple RANSAC (i.e., the process utilized for local registration) is significantly inefficient for global registration, as the GPC contains a sufficiently large number of points. A naive approach to reduce the

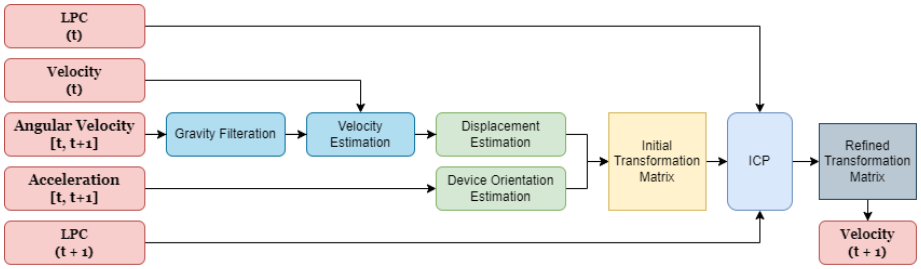


Fig. 10. Inertial-assisted point cloud registration

Algorithm 1 Initial Transformation Matrix Estimation for ICP

function CALCULATE_INITIAL_TRANSFORMATION($timestamps[\,], a[\,][\,], \omega[\,][\,], velocity[\,]$)
  $rotation\_matrix = Identity(3)$
  $translation = [0, 0, 0]$
  for $t \leftarrow 1$ to $timestamps.length - 1$ **do**
    $\Delta t = timestamps[t+1] - timestamps[t]$
    $\Delta r[\,] = \omega[t] * \Delta t$
    $\Delta d[\,] = velocity * \Delta t + \frac{1}{2} * a[t] * \Delta t^2$
    $\Delta d[\,] = rotation\_matrix * \Delta d$
    $rotation\_matrix = CalcRotationMatrix(\Delta r) * rotation\_matrix$
    $translation = translation + \Delta d$
    $velocity = velocity + a[t] * \Delta t$
  end for
  return $translation, rotation\_matrix$
end function

computational overhead, by performing direct voxel downsampling of the point cloud, was observed to result in poor information quality. Instead, we utilize the Open3D implementation of RANSAC, which initially builds a k-d tree (a space partitioning data structure) for fast feature matching and then randomly selects 'n' number of points from the source point cloud. Here, the source point cloud is LPC and the target point cloud is GPC. Such randomly selected points may be distributed across the entire GPC (which is large and has high area coverage) while the LPC corresponds to a smaller region (due to its limited FoV and range). Thus, feature matching performed across these largely distinct point clouds will fail. To solve these challenges, we propose an efficient probabilistic *grid-based RANSAC* approach for a more accurate global registration.

Grid-based RANSAC: In this approach, we first partition the GPC into a grid with equal-sized cells with sufficient overlap (see Figure 11). The partitioning algorithm is presented in Algorithm 2. Let *cell_size* denote half the width or height of a cell. The point cloud is represented in three arrays, each containing x, y, z coordinates. The *FIND_GRID_CENTERS* function calculates the grid centers, while the *PARTITION* function filters the coordinates that fall within each cell. Each cell in the grid is a cube that has a similar number of points as that in LPC. At each instance, RANSAC is executed in parallel for each cell and the cell with highest *fitness score* and *inlier RMSE* (RMSE value of corresponding set) values is selected as the candidate grid, which is then aligned with the LPC (i.e, the source point cloud). The fitness score is defined as the ratio of number of overlapping points[6] to the number of points in the source point cloud. By employing

[6]Two corresponding points are considered to overlap if the distance between them is less than the point cloud voxel size.

this grid-based approach, we achieve a global registration accuracy of 85% (which represents a 12% improvement over the original RANSAC process). However, the execution of RANSAC on every cell increases the registration latency.

Algorithm 2 Grid-RANSAC's Partitioning Algorithm

function FIND_GRID_CENTERS($x[\,], y[\,], z[\,]$)
  $x_{min}, y_{min}, z_{min} \leftarrow MIN(x), MIN(y), MIN(z)$
  $x_{max}, y_{max}, z_{max} \leftarrow MAX(x), MAX(y), MAX(z)$
  $N_x \leftarrow (x_{max} - x_{min})/cell\_size$
  $N_y \leftarrow (y_{max} - y_{min})/cell\_size$
  $z_k \leftarrow (z_{max} + z_{min})/2$
  $centers \leftarrow []$
  for $i \leftarrow 1$ to N_x **do**
    $x_i \leftarrow x_{min} + cell\_size * i$
    for $j \leftarrow 1$ to N_y **do**
      $y_j \leftarrow y_{min} + cell\_size * j$
      $centers.append([x_i, y_j, z_k])$
    end for
  end for
  return *centers*
end function
function PARTITION($x[\,], y[\,], z[\,]$)
  $cx[\,], cy[\,], cz[\,] \leftarrow FIND\_GRID\_CENTERS(x, y, z)$
  $cells \leftarrow []$
  for $i \leftarrow 1$ to $cx.length$ **do**
    $x_p \leftarrow x[x > cx[i] - cell\_size \;\;\&\;\; x < cx[i] + cell\_size]$
    $y_p \leftarrow y[y > cy[i] - cell\_size \;\;\&\;\; y < cy[i] + cell\_size]$
    $z_p \leftarrow z[z > cz[i] - cell\_size \;\;\&\;\; z < cz[i] + cell\_size]$
    $cells.append([x_p, y_p, z_p])$
  end for
  return *cells*
end function

Optimizing latency and accuracy: The grid-based approach partitions the GPC into cells with some overlap. This redundancy is introduced to preserve the information quality of the point cloud (e.g., preventing an object that falls in the boundary of two cells from getting partitioned into two). Also, some of the cells may have either an *insufficient* number of points (e.g., due to lack of enough structural information) or be *redundant* (i.e., almost every point in their region lies in the overlapping region covered by another cell). We empirically determine such cells using a sample feasibility dataset we collected. With this data, we also find the fitness score ranges for correct registration for every cell. We take registrations that fall within a 95% confidence interval (with an upper bound ($mean + 1.96 * SD$) and lower bound ($mean - 1.96 * SD$)) as correct. The upper bound helps to find the over-fitting point clouds in the global registration process. This process effectively removes 41% of the total cells in the grid; this, in turn, improves the global registration accuracy by 9.8% and decreases the registration latency 30-40%. To optimize the latency further, we randomly sample the number of points with their corresponding features (without changing the feature dimensions). Under this randomized sampling (**RS**) optimization, the features are not re-computed for the new point cloud and are thus, more informative than a down sampled-point cloud. The RS optimization improves the global registration latency by 55-60% (e.g., from 2.23secs to 0.95secs for voxel=0.05m), but with an accuracy loss of 11-15%. However, as we shall

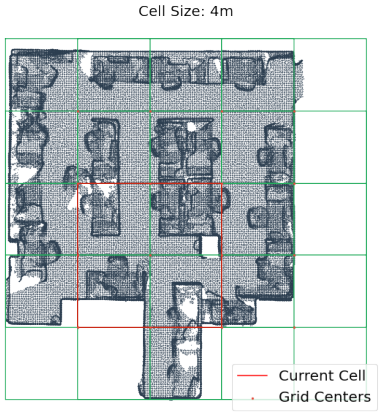


Fig. 11. Partitioned GPC for Grid- based RANSAC

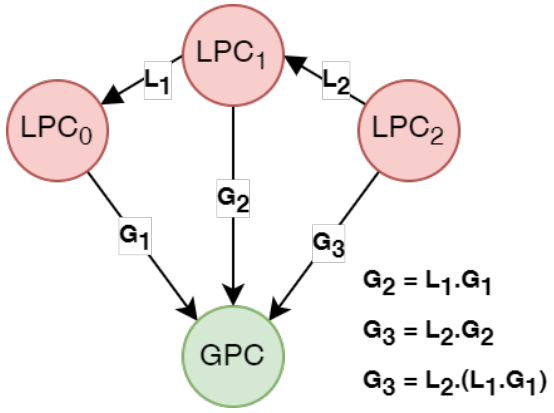


Fig. 12. Global registration verification

describe next (in Section 5.3), this accuracy degradation is palatable as our overall localization process uses multiple such global registrations (for robustness), and is thus able to operate as long as the overall accuracy rate for individual global registrations is ≥66.6%.

5.3 Trajectory Estimation

LiLoc estimates the trajectory traversed by the IoT device by performing local-to-global transformation conversion. To achieve this, ideally one correct global registration is sufficient. But *guaranteeing* such a correct transformation is challenging due to two reasons: (a) lack of any feedback from the user to verify the correctness, and (b) difference in point cloud perspectives (between the global and local LiDAR sensors) and incompleteness of the global point cloud. Because of these reasons, the global registration can sometimes fail even when the inlier RMSE and fitness scores are high. Therefore, relying only on these metrics can lead to poor performance. Accordingly, for *LiLoc*, we devise a more robust, multi-step global *registration and verification* process (illustrated in Figure 13).

5.3.1 Global Registration Verification. To address the above problem, we first compare the transformation matrices computed locally and globally. The GPCs share the same coordinate system, (also referred as the world coordinate system), and the relative motion between two LPCs can be utilized to confirm their registration with corresponding GPCs. The local transformation matrices represent the relative motion between two given time instances. If there are multiple frames in between them, the dot product of their transformation matrices can be taken to find the relative motion. Figure 12 depicts the relationship between the transformation matrices. Let the GPCs at three time instances (t_1, t_2, t_3) be GPC_1, GPC_2, & GPC_3 and let the LPCs be LPC_1, LPC_2, & LPC_3 respectively. As all of them share the same coordinate system, we can consider all of them as one entity GPC. L_1 and L_2 are the transformation matrices (estimated in local registration process) which aligns LPC_2 to LPC_1, and LPC_3 to LPC_2, respectively. The transformation matrix G_i aligns LPC_i and GPC_i. With these transformation matrices, we can build three equality checks: (1) $G_2 = L_1.G_1$, (2) $G_3 = L_2.G_2$, and (3) $G_3 = L_2.(L_1.G_1)$. The equality check is based on the comparison of rotation and translations with two thresholds. The intuition behind an equality check is that there are two ways to register an LPC_i to a GPC_i if we consider the previous global and local transformations. If all the global transformations are correct, all three equality checks will pass. If only two transformations are correct, we can identify the incorrect global registration. However, if

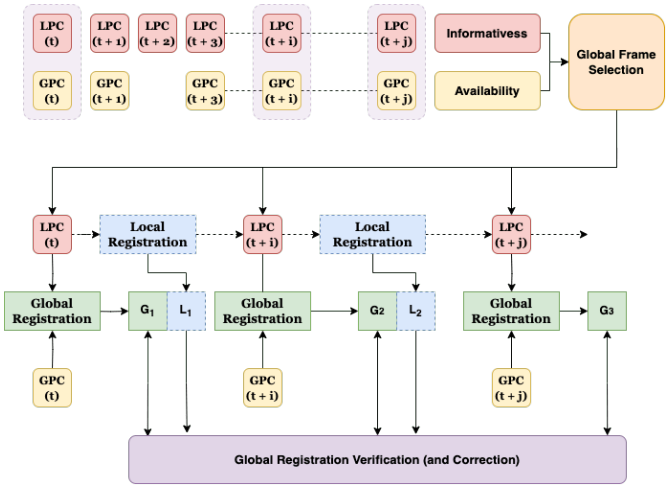


Fig. 13. Global Registration and Validation

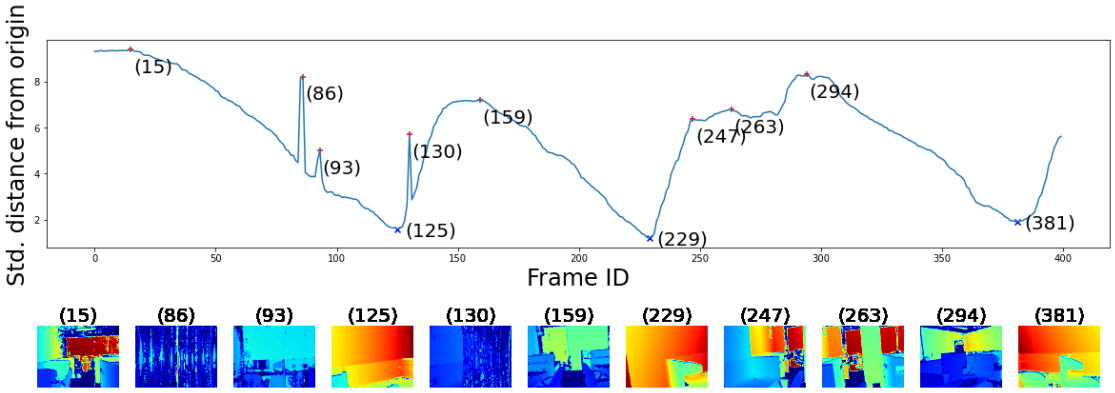


Fig. 14. Identifying non-informative frames

only one of the global transformations is correct, we cannot recognize the correct global registration as all the equality checks will fail.

This process assures both error detection and correction if two conditions are met. First, all the locally estimated transformation matrices have to be correct. Since the local registrations are done with high overlap ratios, the local transformation matrices will likely be correct all the time. Second, there should be a sufficient information difference (i.e., variability across frames) among the three timestamps. We can assure such informativeness through two mechanisms: (a) periodic global registration, whereby successive global registrations are performed with a sufficient gap, which is likely to result in material changes in the GPC (due to the underlying dynamicity of the scene), as well as (b) explicitly measuring the information difference between two point clouds and filtering out *highly-similar* point cloud pairs. The individual steps involved in this verification and correction process are shown in Figure 13.

Identifying Non-Informative Frames: To measure the informativeness of a point cloud, we use a simple metric: the standard deviation of the depth values (*SD_depth*). The depth values are calculated by converting the pixel values of

the depth images to actual depth. Sometimes the depth frames can be noisy due to interference from global LiDAR, occlusions to the local LiDAR camera (e.g., an obstacle blocking the camera) or the frames themself containing no structural information (e.g., frames with only walls). In such instances, we observe either a “sudden" spike or drop in the SD_depth values. Figure 14 shows an example trajectory with such noisy frames identified by sudden spikes (e.g., frame# 86, 130) or drops (e.g., frames# 125, 229) in SD_depth. With an empirically determined threshold, we establish that this approach achieves 75% accuracy in identifying the non-informative frames from the global registration process. This approach also has a secondary benefit: *it provides a marker to indicate when the local registration process is likely to fail, at which point a global registration process must be re-initiated.* In effect, this helps *LiLoc* adopt an *on-demand* global registration paradigm. Additionally, *LiLoc* also mandates an upper time cut-off for (or maximum gap between) global registrations, such that a spot location estimate is obtained even if the local registration process offers meaningful results. This upper bound effectively prevents significant (or unbounded) drift in the trajectory estimation process.

Global Point Cloud Availability: *LiLoc* also takes into account the availability of GPC before performing global registration. Due to potential resource exhaustion in the global LiDAR sensors or transient network outages, certain global frames can be missed. To improve the performance, finding the nearest available global frame is essential. As such, if the time difference between the local frame and the global frame is higher than 100 msec, *LiLoc* refrains from performing global registration.

6 SYSTEM IMPLEMENTATION

We now describe our implementation of the *LiLoc* system. As introduced earlier in Figure 9, our system runs on two separate devices. In our implementation, we utilize an Intel RealSense LiDAR connected to a laptop machine as the secondary device (i.e., the device to be localized) which has AMD Ryzen 7 5800H CPU and 16 GB of RAM. The localization service (which takes input from 3 global LiDARs) runs in a server with GPU (Tesla V100). *LiLoc* is implemented in Python and uses Open3D for processing point clouds and feature matching with RANSAC, and Numpy for other matrix-based computations.

Secondary Device: The local registration process is designed to run on a CPU. We use the Open3D implementation of FPFH to compute the features and RANSAC to obtain the correspondences. As in Open3D implementation, we set the maximum iterations of RANSAC to 300K where only 600 ‘valid’ iterations are enough to stop it. The initial transformation obtained with RANSAC is further refined with Point-to-Point ICP algorithm.

Localization Service: We first stitch point clouds captured from 3 global LiDARs by utilizing a reference point cloud (i.e., a 3D scan of the entire environment) that was captured using an iPad pro LiDAR. Our target is to bring the global cameras’ coordinate systems to a unified coordinate system that is essential for global verification process. We take three point clouds from the global cameras and use FCGF-based RANSAC to get the initial alignment with respect to the reference point cloud. We align them manually if this method fails. To refine the transformation further, we use ICP algorithm.

During global registration, the DNN-based FCGF features are computed using the GPU. We compare two model variants (a 16 and 32-dimension feature vector) that are trained on 3DMatch dataset. The model architecture is ResUNetBN2C and is trained for a 0.025 voxel size. The models are not re-trained on our dataset. Based on our feasibility studies, the model with a 16-dimension feature vector can achieve much better latency than, and comparable accuracy to, the 32-dimension version. Therefore, the FCGF network with 16-dim output is utilized along with RANSAC to compute the initial transformation. The maximum total and validation iterations are set to 4000K and 600, respectively. Point-to-Point ICP algorithm further refines the initial transformation.

7 EVALUATION

In this section, we first explain the experiment setup, dataset collected, set of baselines and evaluation metrics that we use to comprehensively evaluate the performance of our approach. We then evaluate and report the performance of the individual system components of *LiLoc*. Then in Section 7.2 and Section 7.6, we report on the overall system performance in terms of accuracy and latency of localization under real world conditions.

Experiment Setup: To evaluate the performance of *LiLoc*, we conduct several experiments in a multi-occupant university indoor space (with an area of $80m^2$), as shown earlier in Figure 5. We instrument the environment by mounting 3 Intel RealSense L515 LiDAR devices at a height of 3.5m and an angular tilt of 30°. These LiDARs capture point clouds continuously at 8 fps; their captured point clouds are stitched together to obtain the global point cloud (GPC). To obtain the local point cloud (LPC), we instruct multiple people to walk around in the space by carrying a laptop mounted with the realsense LiDAR sensor. The LPC is captured at 30 fps. For inertial-assisted local registration, we use the in-built IMU sensor of the LiDAR sensor which samples device's acceleration and angular velocity at 200 Hz.

Data Collection: To conduct the user studies, we obtain prior approval from our Institutional Review Board (IRB# 21-199-A144(1221)). We collect two datasets to showcase (a) the performance of *LiLoc* in static vs. dynamic environments, and (b) compare the two approaches for local registration. The first dataset *D1* consists primarily of trajectories for a single person (i.e., number of localized devices $N_D = 1$) under two conditions: (a) 10 instances where the person walks in an empty or otherwise unoccupied space, and 17 more trajectories under a *dynamic* setting. To introduce the dynamicity, we vary (a) the number of people simultaneously present/walking in the space from 1 to 5, and (b) the space layout by changing the arrangements of the chairs/tables. We also collect trajectory data when $N_D = 2$, i.e., two participants equipped with LiDAR sensors move around concurrently. Overall, we obtain data for 27 trajectories (10 in static settings & 17 in dynamic environment settings). However, we only use 19 valid trajectories as 8 of them are discarded due to the unavailability of appropriate GPC information. On average, each trajectory is about 20 secs. The second dataset *D2* consists of 30 trajectories for a single individual ($N_D = 1$), but under dynamic conditions where 5 to 7 other individuals are present/walking in the environment. Figure 16 illustrates exemplary instances of dynamic conditions, such as moving individuals obstructing the secondary LiDAR's field of view (Figure 16a), areas without distinguishable features (Figure 16b), and multiple moving individuals within the secondary LiDAR's field of view (Figure 16c). We also alter the layout of the furniture (5-8 chairs, 3-4 tables, 1 whiteboard, and a movable TV) in the room in five different variants during this data collection. Additionally, 2-3 individuals change the position of the chairs and tables during 7 (out of 30) trajectories to simulate a more dynamic environment. Figure 17 further illustrates different furniture layouts and variations in the crowd captured by the global LiDARs. The obtained trajectories include 5 secs calibration time in addition to 20 secs of duration. Only *D2* contains the IMU data along with the LiDAR point clouds. Our dataset is larger, has higher variability and larger FoV compared to the 7-Scenes dataset (with a FoV of 58.5° x 45.6°). Furthermore, we perform an experiment utilizing a moving robotic platform equipped with a LiDAR sensor. As shown in Figure 15, the LiDAR sensor is mounted on the robotic platform at a height of 80 cm. We gather data for 8 trajectories with this robot platform in a static environment where 5 chairs and 4 tables are placed.

Ground Truth Annotation: We also capture the RGB images (from the LiDAR) to help with the manual ground truth annotation process. The precise ground truth for the rotation and translation of the secondary LiDAR cameras are obtained by aligning them manually using the CloudCompare tool [7]. We first split a trajectory into multiple sub-trajectories by identifying the cut-off points based on non-informative frames. These sub-trajectories are registered

[7] https://www.danielgm.net/cc/

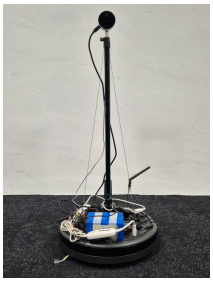
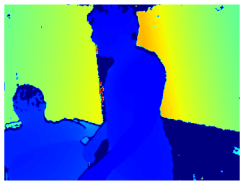
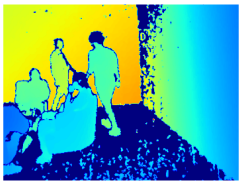


Fig. 15. The robot platform with LiDAR attached to it.



(a) A person obstructing the secondary LiDAR's view

(b) A feature-less area



(c) Multiple moving individuals

Fig. 16. Depth images captured during trajectories, illustrating challenging instances.

Fig. 17. Each column represents depth images captured simultaneously by three global LiDAR sensors during different instances of multiple trajectories. The first two columns demonstrate two different crowd settings in the same layout. The columns 3-6 represent variations in furniture layout and movement of people.

locally as discussed in Section 5.1. For accurate ground truth registrations, the local features are computed using the more heavyweight FCGF instead of FPFH. Each sub-trajectory is verified further by visualizing them. We then manually align these sub-trajectories to a reference global point cloud using CloudCompare tool and refine them with ICP. As mentioned earlier, the reference point cloud is obtained from a separate 3D scan of the environment prior to the start of the experiments, and also used to stitch one global point cloud out of multiple global LIDAR sensors.

Baseline Approaches: We utilize three different baselines:

(a) *Static_NoOpt*: A 'static global point cloud' captured only once at the start of a day and without including any global registration optimization steps.
(b) *Dynamic_NoOpt*: Dynamic global point cloud with 'no optimization' for RANSAC as well as in the selection of global registration.
(c) *Dynamic_NoVer*: Dynamic global point cloud with 'no verification' for global registration.

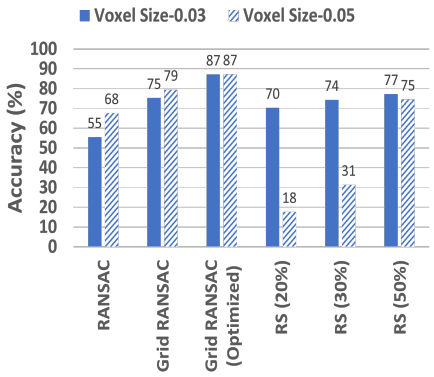


Fig. 18. Global registration accuracy for different approaches

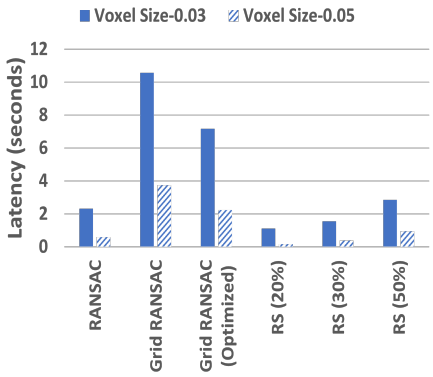


Fig. 19. Global registration latency for different approaches

Evaluation Metrics: To evaluate and compare the performance of *LiLoc* over the baselines, we utilize the following performance metrics:

(a) *Mean Translation Error*: It is computed as the average of the absolute difference between the estimated and ground truth translation vectors along the trajectories
(b) *Mean Rotation Error*: It is the average of the absolute difference between the estimated and ground truth rotation vectors across the trajectories.
(c) *Accuracy of Registration*: The local/global registration accuracy is calculated by averaging the accuracy of pairwise registrations. A pair of point clouds in local registration is two consecutive local point clouds, while a pair of point clouds in global registration is LPC and GPC captured at the same time. A registration of pair is regarded as accurate if the rotational error is less than or equal to 3 degrees, and the translational error is less than or equal to 0.1 meters.
(d) *Accuracy of Trajectory Estimation*: A trajectory is considered accurate if the mean rotational error is less than or equal to 10 degrees, and the mean translational error is less than or equal to 0.3 meters.
(e) *Registration Latency*: The latency for registration is computed as the time required to complete a pair-wise point cloud registration, given only the point clouds as inputs.
(f) *Latency of Localization*: The latency of the *LiLoc* system is defined as the time taken to receive the first global registration result after submitting the first LPC.

7.1 Point Cloud Registration Performance

We first report the global and local point cloud registration performances on our dataset.

7.1.1 Global Registration. **Accuracy**: In Figure 18, we illustrate the accuracy of different variants of global registration on our dataset for point clouds at 0.03 and 0.05 voxel sizes. Our *Optimized Grid-RANSAC* approach achieves more than ~20% performance improvement (with 87% accuracy at both voxel sizes) over basic RANSAC (55.4% and 67.6% accuracy at 0.03 and 0.05 voxels respectively). We further study the registration accuracy for optimized RANSAC with {50%, 30%, 20%} random sampling (RS) of points. We still achieve more than 70% registration accuracy even when only 20% of the points are considered for global registration (at 0.03 voxel). However, there is a significant drop in global registration

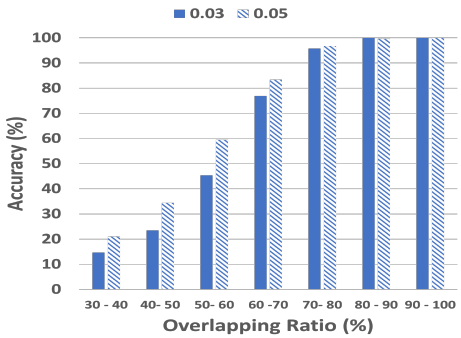


Fig. 20. Local registration accuracy for different voxel sizes with varying overlap ratios

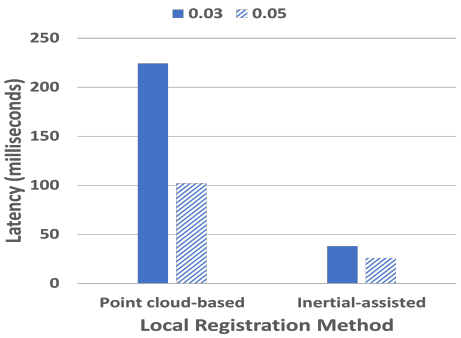


Fig. 21. Local registration latency for different voxel sizes

accuracy for 0.05 voxel size when only 20-30% of the points are sampled. This is because of the significant non-linear reduction in a number of points due to voxel downsampling and random sampling. But the 75% global registration accuracy achieved with 50% random sampling is sufficient for our system. We later show in Section 7.6 that the final trajectory estimation accuracy of *LiLoc* with 50% random sampling remains the same as using all the points.
Latency: We also compare the latency of global registration for different approaches in Figure 19. We achieve significant latency improvement by incorporating random sampling. For *LiLoc* with 50% random sampling, the latency is 2.85 secs and 0.95 secs for voxel sizes 0.03 and 0.05 respectively.

7.1.2 Global LiDAR Coverage and Placement. Suboptimal positioning of global LiDARs can result in an increased frequency of global registrations, imposing a higher computational load on the *Localization Service*. In our experiments, we utilize three global LiDARs (placement as recommended in Section 3.1.4) to generate GPCs. Within dataset *D2*, comprising 30 trajectories, we randomly select 4000 LPCs to assess the hit rate for each global LiDAR. The hit rate of a global LiDAR is defined as the proportion of LPCs that achieve the highest fitness score when registered against the point cloud of the respective global LiDAR, relative to other global LiDARs. The observed hit rates for the three global LiDARs are 34.2%, 34.8%, and 31.0%, respectively. These results suggest that the placement of global LiDARs in our configuration is optimal, as each LiDAR has almost equal hit rates.

7.1.3 Local Registration. **Accuracy**: In Figure 20, we compare the accuracy of point cloud-based local registration for different overlaps (across subsequent LPC frames) and voxel sizes. We observe that nearly perfect local registration accuracy can be achieved when the overlap is at least 70% (which is common for locally captured consecutive frames). The mean accuracy of pair-wise registration for both point cloud-based and IMU-assisted local registration approaches is 99%. However, even one incorrect pair in local registration can affect the overall performance of the trajectory estimation. **Latency**: Similarly, we also perform the latency comparison for the two approaches of local registration (see Figure 21). The latency of point cloud-based registration is directly influenced by the number of points present in the point cloud. The total number of points in a point cloud is determined by the characteristics of the environment and the voxelization process utilized. In this section, we present the latency results for the dataset *D2*, which includes both point cloud and IMU data. For point cloud-based local registration, the latency is approximately 224 msecs for a voxel size of 0.03 m and 102 msecs for a 0.05 m voxel size. On the other hand, when using inertial-assisted local registration,

Table 4. Performance comparison of different local registration methods

Method	Relative Error		Trajectory Error	
	Translation Error (m)	Rotation Error (°)	Translation Error (m)	Rotation Error (°)
Point cloud-based	0.004	0.09	0.311	6.66
Inertial-assisted	0.009	0.19	0.738	14.51
IMU Dead-Reckoning	0.046	1.33	4.898	139.83
RTAB-Map	0.023	0.45	2.583	48.21

the latency significantly reduces to ≈38 msecs (for 0.03 m voxel) and 26 msecs (for 0.05 m voxel). This demonstrates that the inertial-assisted point cloud registration can substantially improve the processing latency, with reductions of 83% and 74.5% for voxel sizes of 0.03 and 0.05, respectively, when compared to the point cloud-based approach. This improvement is achieved by eliminating the FPFH and RANSAC steps involved in the pure point cloud-based approach.

Comparison with RTAB-Map and IMU Dead-reckoning: We compare our approaches for local registration with two other state-of-the-art techniques: (a) RTAB-Map [20], which uses both visual and geometry features to perform SLAM, differs from our approach, which exclusively relies on geometric data and does not incorporate RGB images, and (b) IMU Dead Reckoning [39] in which the trajectory estimation is solely based on accelerometer and gyroscope readings. The trajectories that are sampled from the dataset *D2* for this evaluation have an average length of 14.98 with a standard deviation of 1.02m. Our performance evaluation utilizes two metrics: relative error, which includes the average translation error and the average rotation error between consecutive frames in a trajectory, and trajectory error, which includes the total accumulated translation error and the total accumulated rotation error across the entire trajectory. RTAB-map outputs the pose for a set of keyframes. For a fair comparison, the error values for the other three approaches are also calculated based on these chosen keyframes.

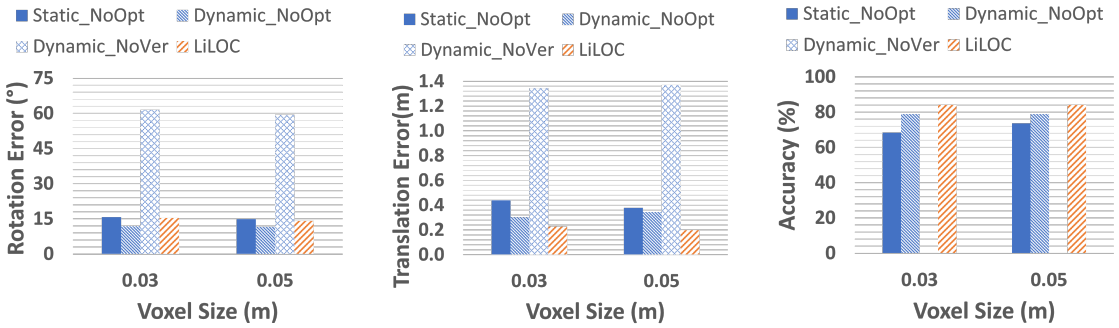
The performance results are tabulated in Table 4. The IMU dead-reckoning approach exhibits the highest error for both relative and trajectory errors, with translation error at 0.046 meters and rotation error at 1.33 degrees. These high error values indicate that the IMU dead-reckoning approach is unsuitable for accurate local registration tasks. The RTAB-Map method has slightly higher translation and rotation errors compared to the point cloud-based and inertial-assisted approaches, with a translation error at 0.023 meters and a rotation error at 0.45 degrees. However, when considering the accumulated error over the entire trajectory, RTAB-Map performs much better than the IMU dead-reckoning approach but is still significantly inferior to the point cloud-based and inertial-assisted approaches. The point cloud-based and inertial-assisted approaches, which utilize a non-informative frame selection technique and have their thresholds determined empirically, achieve significantly better performance on our dataset. We note that our Inertial-assisted approach performs poorer than the pure LPC-GPC registration based approach, but of course, incurs significantly lower bandwidth and computational overheads.

7.2 Localization Performance

We now report the overall performance for continuous location tracking on our dataset *D1* captured under real-world conditions. We compare the mean rotation error (Figure 22a), mean translation error (Figure 22b), and trajectory estimation accuracy (Figure 22c) of *LiLoc* and other baseline approaches. For this comparison, point cloud-based local registration is used. We obtain the difference in translation and rotation values with respect to the ground truth to compute the translation and rotation errors. An estimated trajectory is considered accurate if the mean translation error <= 0.3m and the mean rotation error <= 10°. Using data from 19 valid trajectories, *LiLoc* achieves an accuracy of

Table 5. Global registration performance comparison for different time deadlines

	Global Registration		Local Registration		Accuracy
	Translation (m)	Rotation (°)	Translation (m)	Rotation (°)	(%)
500ms	0.330	19.2	0.034	0.593	78.95
800ms	0.229	15.3	0.046	0.649	84.21
1000ms	0.307	16.9	0.061	0.686	78.95



(a) Rotation Error (b) Translation Error (c) Accuracy

Fig. 22. Continuous location tracking performance (a) rotation error, (b) translation error, and (c) accuracy of *LiLoc* vs baseline approaches **using point cloud-only dataset *D1***.

84.2% with an average rotation error of 14.1° and translation error of 0.198 meters. Out of the 19 trajectories, 3 of them failed due to invalid local registrations. A key reason for the failure of the local registration is the obstructed FoV of the local LiDAR by moving persons or other objects in the environment. When considering only the correctly estimated trajectories, *LiLoc* achieves significantly higher performance with a **rotation error of 3.2°** and **translation error of 0.074 meters**. Note that the *Dynamic_NoVer* variant failed to accurately estimate the final trajectory in all cases due to its higher mean translation and rotation errors exceeding the predefined thresholds (i.e., <= 0.03m and <= 10°), rendering it unsuitable for reliable trajectory estimation.

To study the effect of the cut-off time for global registration on the localization performance, we vary the time deadlines across {500ms, 800ms, 1000ms}. The local registration error increases as the cut-off time for global registration increases, subsequently increasing the global registration error too. Whereas, if global registration is performed more frequently, the difference between point clouds are not sufficient enough to perform verification. We achieve the best performance of 84.2% accuracy, 0.229m translation error and 15.3° rotation error at 800ms as reported in Table 5.

Benefits of removing non-informative frames: We compare the translation and rotation error for correct trajectories in Figure 23 and Figure 24 respectively. When the non-informative frames are discarded, the translation and rotation errors are reduced by 24.9% (from 9.4cm to 7.1cm) and 46.9% (from 5.8° to 3.0°) respectively for 0.03 voxel size. For a 0.05 voxel size, the translation error decreases by 14.7% (from 8.7cm to 7.4cm), but the improvement in rotation error is insignificant.

Static vs. Dynamic Environment: We tabulate the continuous localization performance of *LiLoc* vs. baselines for static vs dynamic environments in Table 6. For this we utilize data from 9 static trajectories (i.e., the environment has only one person moving around) and 10 dynamic trajectories (where there are up to 5 people moving around simultaneously). We observe that for a 0.05 voxel size, mean translation error is 0.178m and 0.216m for static and

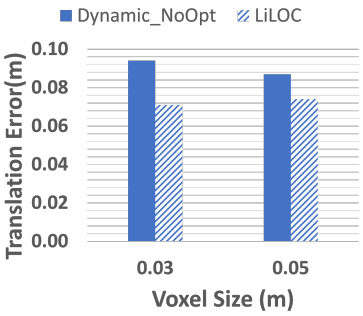


Fig. 23. Translation error for correct trajectories

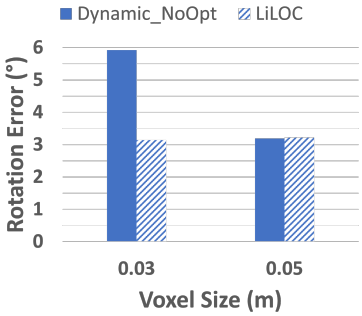
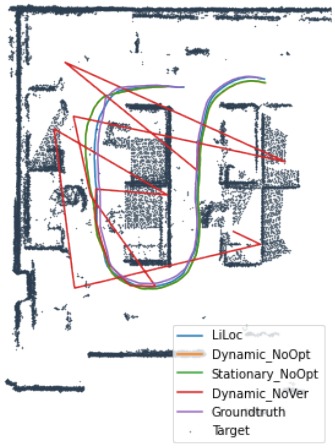
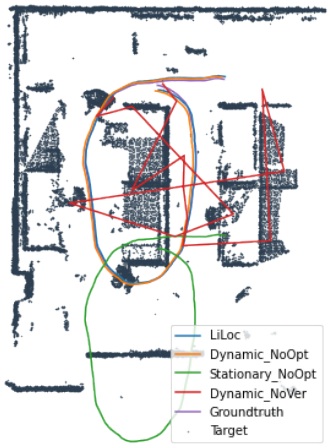


Fig. 24. Rotation Error for correct trajectories



(a) Static Environment



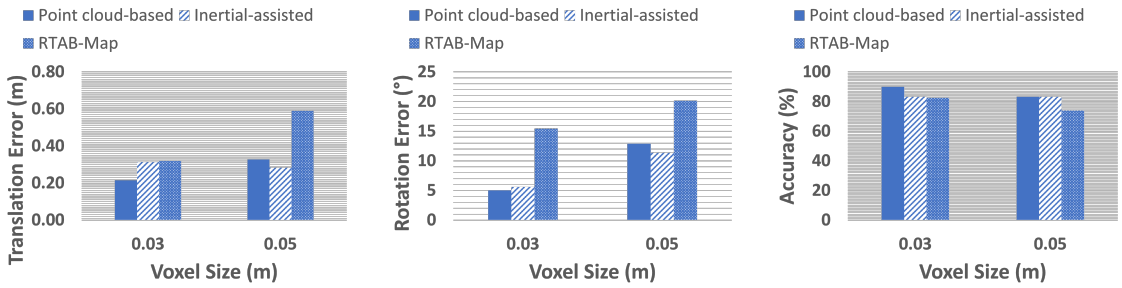
(b) Dynamic Environment

Fig. 25. Example of trajectory estimated by *LiLoc* and baseline approaches vs. ground truth for (a) static, and (b) dynamic environment.

Table 6. Trajectory estimation performance for Empty vs. Crowded environments for different voxel sizes

Voxel Size	Method	Empty Environment			Crowded Environment		
		Avg. Translation Error (m)	Avg. Rotation Error (°)	Accuracy (%)	Avg. Translation Error (m)	Avg. Rotation Error (°)	Accuracy (%)
0.03	Dynamic_NoVer	1.389	55.787	0	1.305	66.666	0
	Static_NoOpt	0.288	7.578	89	0.577	23.197	50
	Dynamic_NoOpt	0.284	6.799	89	0.323	16.395	70
	LiLOC	**0.220**	**11.211**	**89**	**0.237**	**19.129**	**80**
0.05	Dynamic_NoVer	1.488	52.470	0	1.264	65.666	0
	Static_NoOpt	0.425	12.126	77	0.338	17.309	70
	Dynamic_NoOpt	0.298	6.853	89	0.387	15.905	70
	LiLOC	**0.178**	**11.358**	**89**	**0.216**	**16.573**	**80**

dynamic settings respectively (i.e., only a marginal increase in error of 0.038m). Similarly, the rotational error increases marginally from 11.35° to 16.57° and the overall trajectory estimation accuracy drops from 89% to 80%, when the space is crowded. When we consider only the correctly estimated trajectories, the translation error and rotational error drops to **0.068m** and **3.34° even in a dynamic environment**.



(a) Translation Error (b) Rotation Error (c) Accuracy

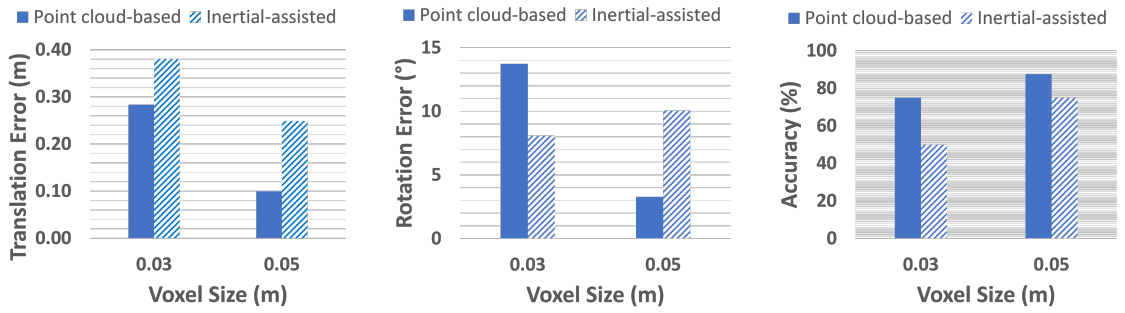
Fig. 26. Comparison of location tracking performance on **point-cloud + IMU Dataset *D2***: (a) translation error, (b) rotation error, and (c) accuracy for the three different local registration approaches.

Figure 25 illustrate two examples of trajectory estimated by *LiLoc* and baselines approaches vs. the ground truth. For a static trajectory (as shown in Figure 25(a)), we can see that *LiLoc* is able to accurately reproduce the trajectory as the original trajectory traversed. In Figure 25(b), we show an example trajectory when the space has 5 people moving around. We can see that the *Static_NoOpt* baseline (i.e., with a static global map) completely fails to recover the trajectory, whereas *LiLoc* is able to precisely estimate the trajectory. The *Static_NoOpt* variant also fails because it does not consider the "informativeness" metric to select the frames and a static global map is used for registration.

7.3 Localization Performance with different local registration approaches

We perform a comparative analysis of rotation error (Figure 26b), translation error (Figure 26a), and trajectory accuracy (Figure 26c) across the three distinct approaches for local registration: point cloud-based, inertial-assisted, and RTAB-Map methods. This assessment is performed on the challenging *D2* dataset, characterized by complex layouts and a higher density of individuals within the space. Notably, trajectory estimation accuracy using point cloud-based local registration reaches 90% with a voxel size of 0.03, showcasing an 8% improvement compared to its performance with a 0.05 voxel size. Moreover, this configuration has a rotation error of 5.03° and a translation error of 0.22m, which are also comparatively better than the rest. This suggests that feature-based point cloud registration with more dense point clouds yields better performance in more dynamic environments compared to inertial-assisted registration. Conversely, both approaches exhibit similar accuracy with the 0.05 voxel size, albeit with slightly higher translation and rotation errors in point cloud-based local registration. The accuracy of *LiLoc* remains consistent for both datasets *D1* and *D2*.

As previously mentioned, RTAB-Map is a state-of-the-art SLAM technique that can be integrated with *LiLoc* to perform local registration. RTAB-Map only outputs the pose for selected keyframes, making a direct comparison with the other two local registration approaches unfair. Unlike the other methods, which perform global registration on a strict schedule, RTAB-Map cannot adhere to a fixed time constraint for global registration. Consequently, global registration and non-informative frame-based trajectory splitting are performed on these keyframes, often resulting in fewer than the required three frames per fragment for global registration verification. As a result, 7 out of 30 trajectories were discarded due to insufficient frames for this task. The accuracy for a 0.03 voxel size using the RTAB-Map method



(a) Translation Error (b) Rotation Error (c) Accuracy

Fig. 27. Comparison of location tracking performance from a **robot's perspective** (a) translation error, (b) rotation error, and (c) accuracy for point cloud-based vs. inertial-assisted local registration process.

is 82.6%, comparable to the other two methods. However, the accuracy drops to 73.9% for the 0.05 voxel size, indicating a slight decline. The translation and rotation errors follow a similar pattern for both voxel sizes.

7.4 Performance of *LiLoc* from a Robot's Perspective

We evaluate *LiLoc's* performance for a LiDAR mounted on a moving robot platform. The robot is controlled manually and is set to move at a constant speed of 15 cm/s. The LiDAR capture point clouds at a similar rate to previous experiments. Due to the slower movement compared to human-carried LiDAR setups, we adjusted the frame rate to 10 frames per second and increased the global registration interval to 1200 milliseconds.

The overall performance results for the data collected with the robot are shown in Figure 27. When comparing the results for *LiLoc*, with data collected from a human perspective (i.e., trajectories from *D*1 in static setting) versus a robot perspective, we observe consistent overall trajectory estimation accuracy on both datasets, approximately at 88%. For the robot's perspective, the translation error shows a slight improvement of 6.4 cm compared to the human perspective, while there is a notable improvement of 8.07° in rotation error. However, it is worth noting that the trajectory length is shorter compared to the trajectories in dataset *D1*.

For the robot-acquired data, Figure 27c presents accuracy results for point cloud-based and inertial-assisted trajectory estimations, demonstrating higher accuracy with a voxel size of 0.05. Furthermore, Figure 27a illustrates translation errors, notably reduced for the 0.05 voxel size compared to 0.03 for both approaches. However, the rotation error for the inertial-assisted approach, shown in Figure 27b, is slightly higher with the 0.05 voxel size. Decreasing the frame rate from 20 to 10 increases the interval between subsequent frames, leading to increased error in the initial transformation matrix estimation for ICP and subsequently affecting the inertial-assisted approach.

7.5 Impact of Integrating Compression Mechanisms

The current implementation of *LiLoc* requires the transfer of at most two point clouds to the server every 800 msecs, which implies a bandwidth overhead and transfer latency of only 4.3 Mbps (2.61 msec) for 0.03 voxel size and 1.7Mbps (1.05 msecs) for 0.05 voxel size. An alternative approach, of offloading local registration entirely to the server will generate prohibitively high volumes of LPC data: Table 7 provides the bandwidth required to transfer local point clouds at 20 fps for two different voxel sizes, and justifies our decision to perform on-device local registration. The low per-device overhead ~1.5-4.5 Mbps implies that *LiLoc*, where the bandwidth requirement increases linearly with an increase in the number of users/devices simultaneously being localized, should be able to support localization

Table 7. Infeasibility of offloaded local registration: LPC transfer bandwidth at 20 fps.

Voxel Size (m)	Average Size of LPC (kB)	Bandwidth for offloading (Mbps)
0.03	277	43.2
0.05	110	17.3

Table 8. Global registration performance with octree-based point cloud compression

Voxel Size	Method	Avg. Translation Error (m)	Avg. Rotation Error (°)	Accuracy (%)
0.03	**GPC + LPC**	0.647	22.277	73
	GPC + Compressed LPC	0.915	33.331	69
0.05	**GPC + LPC**	0.491	17.940	81
	GPC + Compressed LPC	0.587	25.195	80

Table 9. Round-trip time for local point clouds (LPCs)

Voxel Size	RTT (ms)	
	Compressed	Uncompressed
0.03	41	77
0.05	9	35

concurrently for at least 10-20 users. For even greater scalability, we can integrate point cloud compression techniques to reduce the size of LPCs transferred. Recently, octree-based compression, enhanced through the integration of deep learning methods, has emerged as a prominent technique for compressing point clouds [6, 29]. However, deep learning based approaches for compression significantly are significantly more computationally intensive, making them less suitable for real-time localization and navigation tasks. In our implementation, we adopt a straightforward approach to octree-based compression, coupled with voxelization, aimed at reducing point cloud size without imposing a substantial increase in computational overhead. Since our method is a lossy compression scheme, we observe a slight drop in global registration performance for both the voxel sizes (see Table 8). However, this slight drop does not impact the trajectory estimation as the verification process (Section 5.3.1) can compensate for this accuracy drop. Table 9 tabulates the round-trip, measured as the total time needed to transmit an LPC and receive the resulting transformation matrix. It takes <6 msecs to compress on the secondary device and to decompress on the server. This marginal increase in computational latency is more than offset by the decrease in transmission latency, resulting in an overall latency savings of 46 - 70%.

7.6 Localization Latency

The latency of *LiLoc* depends on the user's localization requirement (i.e., spot localization vs. continuous trajectory estimation) and the availability of the IMU sensor in the secondary devices. In Table 10, we report the breakdown of latency of individual components of *LiLoc* for point clouds at voxel sizes 0.03m and 0.05m. For spot localization, *LiLoc* executes global registration once at that time instant without any location verification involved (i.e., same as the *Dynamic_NoVer* baseline). This takes approximately **973 msecs** to compute the FCGF features and execute the optimized grid-RANSAC. If continuous localization is required, the user can switch to trajectory estimation where continual local registration is performed. For the global registration verification process, at least three global registrations are required.

If the deadline is set to 800 msecs, this takes at least 1600 msecs from the first frame to start the verification process. The global registrations for each GPCs are executed in parallel.

The point cloud-based local registration is executed for 16 frames at the secondary device, and it takes about 1632 msecs (i.e., 102 msecs x 16). It takes another 973 msecs to receive the third global registration result. Meanwhile, the secondary device can send the local transformation to the localization service, and it will execute the verification process after 2573 msecs. The verification (i.e., comparison & correction) itself takes only less than 10 msecs. Therefore, the user will receive the first global registration results after 2583 msecs from submitting the first frame.

The inertial-assisted local registration is also executed for 16 frames at the secondary device, but it takes about 416 msecs (i.e., 26 msecs x 16), which is a significant improvement compared to the pure point cloud-based local registration. Similar to the previous approach, it takes 2573 msecs to execute the verification process and 2583 msecs to share the first global registration result. However, after the verification process is completed, the inertial-assisted approach can estimate the trajectory in a real-time fashion as the local registration is significantly faster.

The baselines, *Static_NoOpt* & *Dynamic_NoOpt* also utilize this verification process (similar to *LiLoc*), and thus have a similar lower bound for their latency. Figure 28 depicts the execution time for different components of *LiLoc* vs baselines.

	Voxel Size	
	0.03	**0.05**
Local Registration (ms)	38/224	26/102
Global Registration (ms)	2878	973
Trajectory Estimation (ms)	<10	<10

Table 10. Latency for different voxel sizes

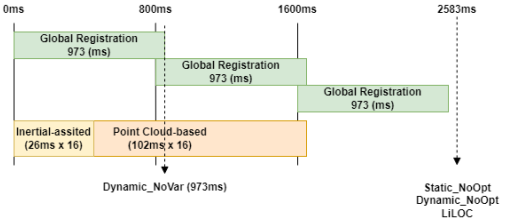


Fig. 28. Latency for different approaches(voxel size=0.05)

7.7 Energy Overhead

The secondary device consumes power for two main tasks: capturing point clouds and local registration. We measure the CPU wattage for the local registration process: the point cloud-based approach consumes ≈ 58.46W, whereas the inertial-assisted approach consumes ≈ 25.24W. The LiDAR sensor itself consumes 4.26W and remains always-on under both approaches, while the IMU sensor consumes an additional 1.62W. Nonetheless, even with the additional overhead of IMU sensing, the inertial-assisted approach has a 50+% lower power overhead as it avoids the computational overheads of feature computation and RANSAC-based feature matching. Additionally, the WiFi module consumes ≈ 1.3W to transmit point clouds wirelessly to the *Localization Server*.

8 DISCUSSION

Scalability to larger crowds. To study the sensitivity of our approach in scaling to more-crowded dynamic environments, we conducted an experiment with 12 people simultaneously present/moving within our recently-renovated experimental lab space, with a diverse set of furniture layouts. We observed that *LiLoc* can accurately track the trajectories with an average translation error of 0.16m, rotation error of 7.3° and a trajectory estimation accuracy of 76.9% (10 out of 13 trajectories are correctly estimated). The slight drop in accuracy (compared to 84% accuracy achieved with only 1-5 people) is mainly due to failure of local registrations as often the local frames are occluded by parts of

the moving subjects. Overall, we infer that *LiLoc's* (location, pose) estimation remains highly accurate even in more crowded settings, although there is a slightly higher chance of unsuccessful instantaneous localization.

Improving local registration. The process of local registration, which executes on the mobile platform as per the *LiLoc* architecture, takes ∼134ms, including the steps of informativeness calculation, feature extraction, and registration. For fine-grained continuous tracking of human-like indoor movement (speed=∼1.5m/sec), given the ∼10cm margin of error, this latency should ideally be $\frac{10}{150} * 1000 = \sim 67$ms. Besides natural organic growth in hardware capabilities, one possible 'straightforward' way to reduce the latency is to incorporate alternative lightweight trajectory estimation techniques, such as inertial-sensing-based dead reckoning. However, such techniques are known to be susceptible to certain environmental perturbations (e.g., ferromagnetic influence near escalators and lifts) and should thus be combined judiciously with LiDAR-based registration to achieve even lower-latency, accurate tracking. In fact, in our own experiments, pure IMU-based dead reckoning had a displacement error up to 1.6 m over a 5 sec interval. As this is considerably higher than LILOC's pose estimation error, we instead utilized inertial sensing purely as a means to obtain coarse-grained transformation matrix estimates.

Implementing an adaptive LiDAR scanning strategy The current Intel RealSense LiDAR does not allow to set a variable frame rate for the LiDAR scanner, thereby consuming a constant power all the time. However, it allows for hardware synchronization through a signalling port, which takes about 121msecs to receive the first depth frame after it is triggered with the hardware sync signal. This currently impedes the possibility of implementing an adaptive LiDAR scanning strategy that might complement the inertial sensing-based dead-reckoning by setting a variable frame rate.

RGB-to-point cloud correspondence. It is perhaps natural to ponder whether *LiLoc* can be modified to operate without requiring LiDAR sensors on the mobile/IoT device. Conceptually, RGB images can be processed by image-to-depth estimation DNNs to indirectly generate the LPC, even though executing such DNNs will require hardware accelerators (e.g., GPU, NPU) that may not be universally available on pervasive devices. In addition, the current state-of-the-art image-to-depth models suffer from a depth estimation error of ∼30cm per pixel, which in turn results in poor registration performance. Therefore, this functionality remains as a future work.

Position of Local LiDAR: In our current experimental studies, the local LiDAR sensor is always located at a height of about 1 meter from the floor. However, certain smart objects may have the LiDAR mounted at a lower height–e.g., close to ground level in a robot vacuum cleaner. Additional studies are needed to ascertain whether the LPCs captured from such near-ground perspectives are informative enough to permit successful (LPC, GPC) registration and device localization.

9 CONCLUSIONS

We have demonstrated the feasibility and performance of *LiLoc*, a multi-perspective LiDAR-generated point cloud-based continuous localization approach for real-world dynamic indoor environments. *LiLoc's* key innovations include (a) a combination of optimized grid-based RANSAC with random sampling for achieving accurate, intermittent spot location estimates (at low latency), (b) a light-weight approach using point cloud registration across local frames to capture relative motion during the intermediate time periods and (c) a further optimization of the local registration process, using inertial sensing to bypass computationally expensive steps of feature computation and RANSAC-based matching. Experimental results show that *LiLoc* achieves: (a) fine-grained localization performance with an average translation error of only 7.4 cms and average rotation error of 3.2°, (b) significantly better performance (12% improvement) under dynamically changing environment conditions when compared to an approach that uses a quasi-static global point cloud, and (c) a spot localization latency of only 973 msecs. Moreover, by judiciously interleaving inertial sensing with

point cloud-based local registration, *LiLoc* can reduce registration latency and power by ≈80% and 50+%, respectively and by integrating compression mechanism, we achieve about 46% reduction in the network transmission latency.

ACKNOWLEDGMENTS

This work was supported by National Research Foundation, Prime Minister's Office, Singapore, under its NRF Investigatorship grant (NRF-NRFI05-2019-0007), and partially under its Campus for Research Excellence and Technological Enterprise (CREATE) program. The Mens, Manus, and Machina (M3S) is an interdisciplinary research group (IRG) of the Singapore MIT Alliance for Research and Technology (SMART) centre. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Paramvir Bahl and Venkata N Padmanabhan. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proc. IEEE INFOCOM 2000*, Vol. 2. Ieee, 775–784.

[2] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. 2020. D3feat: Joint learning of dense detection and description of 3d local features. In *Proc. of the IEEE/CVF CVPR'2020*.

[3] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. Spie, 586–606.

[4] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. 2002. The trimmed iterative closest point algorithm. In *Proc. of ICPR'02*.

[5] Christopher Choy, Jaesik Park, and Vladlen Koltun. 2019. Fully convolutional geometric features. In *Proc. of the IEEE/CVF ICCV'2019*.

[6] Mingyue Cui, Junhua Long, Mingjian Feng, Boyang Li, and Huang Kai. 2023. OctFormer: Efficient octree-based transformer for point cloud compression with local enhancement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 470–478.

[7] Paolo Dabove, Vincenzo Di Pietra, Marco Piras, Ansar Abdul Jabbar, and Syed Ali Kazim. 2018. Indoor positioning using Ultra-wide band (UWB) technologies: Positioning accuracies and sensors' performances. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 175–184.

[8] Mahdi Elhousni and Xinming Huang. 2020. A survey on 3d lidar localization for autonomous vehicles. In *Proc. of IEEE Intelligent Vehicles Symposium*.

[9] Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[10] Shiyi Guo, Zheng Rong, Shuo Wang, and Yihong Wu. 2022. A LiDAR SLAM With PCA-Based Feature Extraction and Two-Stage Matching. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–11.

[11] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. 2016. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 1271–1278.

[12] Xin Hu, Lianglun Cheng, and Guangchi Zhang. 2011. A Zigbee-based localization algorithm for indoor environments. In *Proc. of ICCN'11*, Vol. 3. IEEE, 1776–1781.

[13] Beakcheol Jang and Hyunjung Kim. 2018. Indoor positioning technologies without offline fingerprinting map: A survey. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 508–525.

[14] Kaijin Ji, Huiyan Chen, Huijun Di, Jianwei Gong, Guangming Xiong, Jianyong Qi, and Tao Yi. 2018. CPFG-SLAM: A robust simultaneous localization and mapping based on LIDAR in off-road environment. In *Proc. of IEEE Intelligent Vehicles Symposium (IV)*.

[15] Xingliang Ji, Lin Zuo, Changhua Zhang, and Yu Liu. 2019. Lloam: Lidar odometry and mapping with loop-closure detection based correction. In *Proc. of IEEE ICMA'19*.

[16] Hong Jiang, Chao Peng, and Jing Sun. 2019. Deep belief network for fingerprinting-based RFID indoor localization. In *Proc. of ICC'19*. IEEE, 1–5.

[17] Wonho Kang and Youngnam Han. 2014. SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors journal* 15, 5 (2014).

[18] Kenji Koide, Jun Miura, and Emanuele Menegatti. 2019. A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement. *International Journal of Advanced Robotic Systems* 16, 2 (2019), 1729881419841532.

[19] Toshiaki Koike-Akino, Pu Wang, Milutin Pajovic, Haijian Sun, and Philip V Orlik. 2020. Fingerprinting-based indoor localization with commercial MMWave WiFi: A deep learning approach. *IEEE Access* 8 (2020), 84879–84892.

[20] Mathieu Labbé and François Michaud. 2019. 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* 36, 2 (2019), 416–446.

[21] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. 2014. Epsilon: A visible light based positioning system. In *Proc. of USENIX NSDI'14*.

[22] Jongil Lim, SeokJu Lee, Girma Tewolde, and Jaerock Kwon. 2015. Indoor localization and navigation for a mobile robot equipped with rotating ultrasonic sensors using a smartphone as the robot's brain. In *Proc. of IEEE International Conference on Electro/Information Technology (EIT)*. IEEE.

[23] Huayao Liu, Ruiping Liu, Kailun Yang, Jiaming Zhang, Kunyu Peng, and Rainer Stiefelhagen. 2021. HIDA: Towards holistic indoor understanding for the visually impaired via semantic instance segmentation with a wearable solid-state LiDAR sensor. In *Proc.of the IEEE/CVF ICCV'2021*.

[24] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. 2019. Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proc.of the IEEE/CVF CVPR'19*.

[25] Kazuki Misu and Jun Miura. 2015. Specific person tracking using 3D LIDAR and ESPAR antenna for mobile service robots. *Advanced Robotics* 29, 22 (2015), 1483–1495.

[26] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics* 31, 5 (2015), 1147–1163.

[27] Thien-Minh Nguyen, Zhirong Qiu, Thien Hoang Nguyen, Muqing Cao, and Lihua Xie. 2019. Persistently excited adaptive relative localization and time-varying formation of robot swarms. *IEEE Transactions on Robotics* 36, 2 (2019), 553–560.

[28] Kenji Omasa, Fumiki Hosoi, and Atsumi Konishi. 2007. 3D lidar imaging for detecting and understanding plant responses and canopy structure. *Journal of experimental botany* 58, 4 (2007), 881–898.

[29] Z. Que, G. Lu, and D. Xu. 2021. VoxelContext-Net: An Octree based Framework for Point Cloud Compression. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 6038–6047.

[30] Hazem Rashed, Mohamed Ramzy, Victor Vaquero, Ahmad El Sallab, Ganesh Sistu, and Senthil Yogamani. 2019. Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving. In *Proc. of IEEE/CVF ICCV'19*.

[31] Jenny Röbesaat, Peilin Zhang, Mohamed Abdelaal, and Oliver Theel. 2017. An improved BLE indoor localization with Kalman-based fusion: An experimental study. *Sensors* 17, 5 (2017), 951.

[32] Dávid Rozenberszki and András L Majdik. 2020. LOL: Lidar-only odometry and localization in 3D point cloud maps. In *Proc. of IEEE ICRA'2020*.

[33] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. 2009. Fast point feature histograms (FPFH) for 3D registration. In *Proc. of IEEE ICRA'09*.

[34] Sriram Sami, Yimin Dai, Sean Rui Xiang Tan, Nirupam Roy, and Jun Han. 2020. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors. In *Proc. of ACM SenSys'2020*.

[35] Jukka Talvitie, Markku Renfors, and Elena Simona Lohan. 2015. Distance-based interpolation and extrapolation methods for RSS-based localization with indoor wireless signals. *IEEE transactions on vehicular technology* 64, 4 (2015), 1340–1353.

[36] Martin Velas, Michal Spanel, and Adam Herout. 2016. Collar line segments for fast odometry estimation from velodyne point clouds. In *Proc. of IEEE ICRA'16*.

[37] Han Wang, Chen Wang, and Lihua Xie. 2021. Lightweight 3-D localization and mapping for solid-state LiDAR. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1801–1807.

[38] T.B. Welch, R.L. Musselman, B.A. Emessiene, P.D. Gift, D.K. Choudhury, D.N. Cassadine, and S.M. Yano. 2002. The effects of the human body on UWB signal propagation in an indoor environment. *IEEE Journal on Selected Areas in Communications* 20, 9 (2002), 1778–1782.

[39] Oliver J Woodman. 2007. *An introduction to inertial navigation*. Technical Report. University of Cambridge, Computer Laboratory.

[40] Chenshu Wu, Zheng Yang, and Yunhao Liu. 2014. Smartphones based crowdsourcing for indoor localization. *IEEE Transactions on Mobile Computing* 14, 2 (2014), 444–457.

[41] Jie Yang and Yingying Chen. 2009. Indoor localization using improved rss-based lateration methods. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 1–6.

[42] Jianfei Yang, Han Zou, Hao Jiang, and Lihua Xie. 2018. Device-free occupant activity sensing using WiFi-enabled IoT devices for smart homes. *IEEE Internet of Things Journal* 5, 5 (2018), 3991–4002.

[43] Zheng Yang, Chenshu Wu, and Yunhao Liu. 2012. Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Proc. of MobiCom'12*. 269–280.

[44] Huan Yin, Li Tang, Xiaqing Ding, Yue Wang, and Rong Xiong. 2018. Locnet: Global localization in 3d point clouds for mobile vehicles. In *Proc. of IEEE Intelligent Vehicles Symposium (IV)*.

[45] Moustafa Youssef and Ashok Agrawala. 2005. The Horus WLAN location determination system. In *Proc. of ACM MobiSys'05*.

[46] Chi Zhang and Xinyu Zhang. 2017. Pulsar: Towards ubiquitous visible light localization. In *Proc. of ACM MobiCom'17*.

[47] Ji Zhang and Sanjiv Singh. 2015. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Proc. of IEEE ICRA'15*.

[48] Minghui Zhao, Tyler Chang, Aditya Arun, Roshan Ayyalasomayajula, Chi Zhang, and Dinesh Bharadia. 2021. Uloc: Low-power, scalable and cm-accurate uwb-tag localization and tracking for indoor applications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–31.

[49] Bo Zhou, Zhongqiang Tang, Kun Qian, Fang Fang, and Xudong Ma. 2017. A lidar odometry for outdoor mobile robots using ndt based scan matching in gps-denied environments. In *Proc. of IEEE CYBER'17*. IEEE.

[50] Han Zou, Ming Jin, Hao Jiang, Lihua Xie, and Costas J Spanos. 2017. WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation. *IEEE Transactions on Wireless Communications* 16, 12 (2017), 8118–8130.