

D-LIO: 6DoF Direct LiDAR-Inertial Odometry Based on Simultaneous Truncated Distance Field Mapping

L. Coto-Elena ¹, J.E. Maese ¹, L. Merino ¹, *Member, IEEE*, and F. Caballero ¹

Abstract—This letter presents a new approach for 6DoF Direct LiDAR-Inertial Odometry (D-LIO) based on the simultaneous mapping of truncated distance fields on CPU. Such continuous representation (in the vicinity of the points) enables working with raw 3D LiDAR data online, avoiding the need of LiDAR feature selection and tracking, simplifying the odometry pipeline and easily generalizing to many scenarios. The method is based on the proposed Fast Truncated Distance Field (Fast-TDF) method as a convenient tool to represent the environment, employing binary masks that encodes the L1 distance. Such representation enables i) solving the LiDAR point-cloud registration as a nonlinear optimization process without the need of selecting/tracking LiDAR features in the input data, ii) simultaneously producing an accurate truncated distance field map of the environment, and iii) updating such map at constant time independently of its size. The approach is tested using open datasets, aerial and ground. It is also benchmarked against other state-of-the-art odometry approaches, demonstrating the same or better level of accuracy with the added value of an online-generated TDF representation of the environment, that can be used for other robotics tasks as planning or collision avoidance.

Index Terms—Mapping, odometry, simultaneous localization and mapping.

I. INTRODUCTION

ACCURATE vehicle localization is a crucial aspect of robotics, directly influencing autonomous navigation, remote exploration, and other advanced applications. Various techniques are employed to improve localization, combining data from different sensors such as cameras, inertial measurement units (IMUs), LiDAR and radar [1]. LiDAR-based approaches [2] are widely used due to their capacity to provide high-density and accurate 3D data unaffected by lighting conditions. These techniques have been proven

Received 22 May 2025; accepted 26 October 2025. Date of publication 14 November 2025; date of current version 19 November 2025. This article was recommended for publication by Associate Editor Brendan Englot and Editor Javier Civera upon evaluation of the reviewers' comments. This work was supported in part by INSERTION funded by the "Agencia Estatal de Investigación – Ministerio de Ciencia, Innovación y Universidades" and the "European Union NextGenerationEU/PRTR" under Grant PID2021-127648OB-C31 and in part by NORDIC funded by the "Agencia Estatal de Investigación – Ministerio de Ciencia, Innovación y Universidades" and the "European Union NextGenerationEU/PRTR" under Grant TED2021-132476B-I00. (*Corresponding author: L. Coto-Elena.*)

The authors are with the Service Robotics Laboratory, Universidad Pablo de Olavide, 41013 Seville, Spain (e-mail: lcotele@upo.es; jemaeealv@upo.es; lmercab@upo.es; fcaballero@upo.es).

The source code is publicly available at <https://github.com/robotics-upo/D-LIO.git>.

Digital Object Identifier 10.1109/LRA.2025.3632615

to provide accurate positioning while also enabling environmental mapping [3][4]. However, they have certain limitations. In feature-sparse environments, traditional feature-based techniques [5][6] may struggle to extract reliable information, leading to suboptimal localization. Additionally, when dealing with extremely dense point clouds, the computational cost increases significantly, making real-time execution challenging.

Beyond the choice of sensors, the underlying map representation plays a pivotal role in the accuracy and efficiency of localization systems. Most of the state-of-the-art LiDAR SLAM approaches represent maps in the form of point clouds (3D points and/or features) together with ICP-like methods for LiDAR registration [7][2]. While effective, these discrete representations often face challenges in scalability, sparsity, and differentiability [8][9]. To address these limitations, distance fields, particularly Signed and Truncated Distance Fields [10], have emerged as a promising alternative. These volumetric representations capture obstacle proximity in a continuous fashion, which is especially beneficial for point cloud registration tasks by providing smooth gradients and implicit surface descriptions. However, the application to LiDAR SLAM is still limited to relatively small volumes (rooms, house floor, etc) or with large quantization errors that prevent their use in general purpose application [11][12]. As a result, most LiDAR Distance Field approaches in the state of the art focus on mapping, assuming perfect sensor localization [13][8].

This letter presents, to the best of our knowledge, the first 6DoF LiDAR-Inertial odometry method based on simultaneous mapping of truncated distance field in large volumes. Although the approach has potential for parallelization, it is able to map the environment and estimate the odometry approaching real time using just CPU. Unlike learning-based methods, the proposed Fast-TDF approach provides an accurate representation of the environment from the first scan.

The paper is structured as follows. First, Section II analyzes the current state of the art on LiDAR odometry and distance field computation. Later, the proposed Fast Truncated Distance Field method is presented in Section III. Then, Section IV details the LiDAR-Inertial odometry approach based on simultaneous TDF mapping. The proposed methods are tested and benchmarked using public datasets and open-sourced LiDAR odometry approaches in Section V. Finally, Section VI presents the conclusions and future work.

II. STATE OF THE ART

A. LiDAR-Based Odometry

Modern LiDAR-based odometry relies on high-resolution 3D point clouds to estimate motion, offering a key advantage over cameras by operating independently of ambient lighting conditions. Traditional methods, such as ICP [14] and NDT [15], register consecutive scans by minimizing alignment errors. These have been enhanced by variants like GICP [7], which boost robustness and efficiency. Advanced methods such as CT-ICP [16], KISS-ICP [2], DLO [4] and DLIO [17] all perform scan-to-local-map (or scan-to-submap) registration, leveraging accumulated map information for improved stability. To reduce computational cost, feature-based methods extract geometric structures rather than using entire point clouds. LOAM [3] pioneered this approach by detecting edges and planar features, later extended by LeGO-LOAM [5], F-LOAM [6], and A-LOAM [18], while M-LOAM [19] adapts the framework to multi-LiDAR configurations. In contrast, SuMa++ [20] abandons sparse features in favor of a dense surfel map, each surfel encoding position, normal, temporal stability, and semantic information, for projective ICP registration and dynamic-object filtering.

Deep learning-based odometry has emerged to address challenges in feature consistency and matching. LO-Net [21] improves scan alignment by predicting surface normals and enforcing geometric constraints, while LodoNet [22] uses keypoint-based pose estimation. Unsupervised learning techniques, such as VertexNet [23], model pose uncertainty to enhance accuracy across diverse environments. These methods aim to improve generalization and robustness, complementing traditional LiDAR techniques.

Despite its advantages, LiDAR-only odometry faces several limitations. In featureless environments, such as tunnels, point cloud registration becomes unreliable due to insufficient constraints. Additionally, LiDAR's low frame rate makes it less effective for capturing rapid motion dynamics. To mitigate these issues, sensor fusion with complementary modalities is employed to enhance robustness and computational efficiency. LiDAR-Inertial odometry is an approach commonly used as IMU provide high-frequency inertial measurements capturing short-term motion dynamics with minimal latency.

Sensor fusion methods can be categorized depending on how each sensor contributes to the estimation of the pose. In loosely coupled approaches, LiDAR and IMU measurements are processed independently, and their outputs are fused at a later stage. These methods offer modularity and robustness, but may not fully exploit the complementary nature of both sensors. Examples include LOAM [3] or Lego-LOAM [5].

On the other hand, tightly coupled approaches directly integrate raw IMU and LiDAR measurements within the same optimization framework, enabling continuous state estimation and higher accuracy. Methods such as LIO-SAM [1], FastLIO2 [24] and Faster-LIO [25] perform well in feature-sparse environments, where IMU constraints help maintain reliable pose estimates. Despite advances in LiDAR-based odometry, several challenges still hinder its performance. One of the main issues is the high computational cost of processing dense point clouds,

making it difficult to execute registration and optimization algorithms efficiently on hardware-constrained platforms. Another significant challenge is the difficulty of extracting discriminative features in homogeneous environments or repetitive geometries, potentially leading to errors in motion estimation. In such scenarios, direct odometry methods, that leverage the full point cloud information without relying on feature extraction, are often considered as an alternative [4] [20]. Furthermore, filtering and data reduction strategies must balance the need to decrease computational load while preserving essential information for localization. Recently, neural implicit representations have been explored as an alternative to traditional geometric or TDF-based maps. Approaches such as NeRF-LOAM [26], PIN-SLAM [27], and KN-LIO [28] jointly optimize odometry and dense mapping within continuous neural fields. These methods leverage signed distance functions or neural point-based representations to achieve accurate pose estimation together with high-fidelity 3D reconstructions, often integrating LiDAR and IMU measurements. Although promising, they typically involve high computational demands and GPU acceleration, which may limit their applicability in resource-constrained robotic platforms.

B. Distance Fields Computation

Volumetric mapping and SLAM methods primarily utilize discrete voxel grids, hierarchical data structures, probabilistic models, and more recently implicit neural fields.

Voxel-based methods are extensively used due to their simplicity and ease of implementation. Voxblox [29] efficiently generates Euclidean Signed Distance Fields (ESDF) from Truncated Signed Distance Fields (TSDF), establishing a common baseline in volumetric mapping. Additionally, Voxfield [8] refines accuracy through non-projective TSDF fusion complemented by normal estimation from input point-cloud data. Notably, both Voxblox and Voxfield use voxel hashing, which enables sparse spatial memory allocation and helps mitigate scalability issues in large-scale environments. However, these approaches can still suffer from discretization artifacts and access-time variability, particularly in CPU-only systems or high-frequency mapping tasks.

Hierarchical data structures such as OpenVDB [9] have been proposed to mitigate scalability constraints. Approaches leveraging OpenVDB include VDBFusion [30], which integrates TSDF data into OpenVDB. While these hierarchical methods improve scalability and memory usage, their discrete nature still limits smooth interpolation capabilities and continuous optimization.

Implicit neural representations have emerged as continuous and differentiable alternatives to discrete volumetric maps. DeepSDF [31] introduced neural SDFs for continuous surface modeling, and real-time incremental updates were first achieved by iSDF [11]. More recently, HIO-SDF [12] combines global voxel structures with local LiDAR data for online training, while SHINE-Mapping [32] employs a sparse-octree neural SDF for large-scale reconstruction, but it targets mapping rather than online LiDAR-inertial odometry. NeRF-SLAM [33] delivers real-time dense monocular SLAM with hierarchical NeRFs, yet

none of them unifies full 6-DoF LiDAR-inertial odometry with on-the-fly distance field mapping on a CPU. Despite these advances, neural methods typically require significant pre-training, heavy computation, and are limited to small-scale SLAM due to memory and inference constraints.

Gaussian Processes (GP) have been proposed as probabilistic distance-field representations with built-in uncertainty estimation. GP-based mapping methods remain computationally expensive due to covariance matrix inversion, even when hierarchical structures such as Octrees are used. VDB-GPDF [13] embeds logarithmic GP models in OpenVDB to produce continuous ESDFs with uncertainty, but GP inference still limits real-time performance.

A significant limitation among these methodologies is their primary orientation toward mapping tasks, with limited or no direct application to real-time SLAM due to computational demands and quantization errors inherent in discretized representations. Consequently, real-time trajectory estimation and incremental map updates remain challenging.

In contrast, the approach proposed in this letter, Direct LiDAR-Inertial Odometry (D-LIO), explicitly targets these limitations. D-LIO simplifies the odometry pipeline by eliminating the necessity for explicit LiDAR feature selection and tracking. Moreover, it maintains the environment map update process in constant time, irrespective of map size, significantly enhancing scalability. By directly fusing raw LiDAR point-cloud data into an optimized Fast Truncated Distance Field (Fast-TDF) representation, our method eliminates the additional computational overhead typically required to convert TSDF to ESDF, thus facilitating real-time integration into SLAM systems.

III. FAST TRUNCATED DISTANCE FIELD

Distance Fields (DFs) are able to provide the distance to the closest obstacle from any point in space, no matter how far we are from the obstacle. On the other hand, Truncated Distance Fields (TDFs) are only useful when the coordinate has an obstacle under its truncation distance. While this might be seen as a great advantage of DFs for localization, practical aspects limits and even dismiss such advantage because far points are normally rejected by the outlier rejection process during point cloud registration. Thus, if the truncation distance is properly tuned according to the localization approach at hand, TDFs might be a very efficient representation of the distance field.

This letter proposes using TDF to estimate the distance field. The proposed Fast-TDF takes advantage of binary masks to compute the L_1 distance into the area of influence of each obstacle. We will see how using binary masks to encode TDFs is very convenient, changing the computational paradigm from map size dependence to point cloud size dependence, so that the computation will depend on the number of points to be included into the map, not the map size.

A. Binary Kernel

Our Fast-TDF makes use of a binary kernel to compute the distance. This kernel must be applied to the map in each occupied cell. As a result, it will produce the distance of the

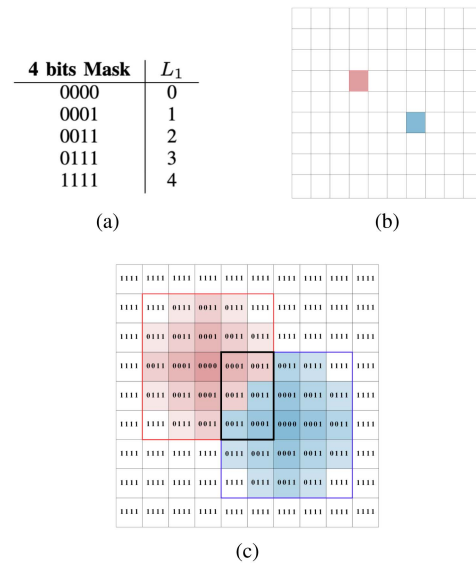


Fig. 1. 2D example of 5×5 binary kernel using 4 bits encoding. (a) Correspondence between mask and L_1 distance. (b) 2D occupancy representation of the input point cloud over the grid. (c) Merged distance map resulting from the bitwise-AND operation over overlapping binary kernels centered at each occupied cell.

surrounding free cells to the closest occupied under the truncation distance. The kernel represents the L_1 distance from each cell to an occupied cell in the center of the kernel. The distances into the kernel are expressed as bit-set masks following these conventions (Fig. 1(a) shows an example of four bits):

- A zeroed mask indicates L_1 distance equal to zero.
- Starting with the Least Significant Bit (LSB), each bit set accounts for the distance between two consecutive cells.
- An all-one mask indicates the maximum L_1 distance, which is equal to the number of bits in the mask.

This distance encoding has a great advantage, we can compute the shortest distance into an arbitrary set by means of the bitwise-AND operation. No matter the number of distances or the values involved, a bitwise-AND among all the distances will provide the shortest value. This is positive at three levels: i) The bitwise-AND operation can be computed extremely fast in computers, with both, scalar and vectorized instructions, ii) We do not need to perform value comparisons at all, with the corresponding computation impact, and iii) The distance insertion order does not affect the result, because the bitwise-AND operation is commutative, which simplifies the parallelism.

As an illustrative example, let's assume we have three different distance values represented as a bit-set mask of 8 bits, these values are 00111111 ($L_1 = 6$), 00001111 ($L_1 = 4$) and 00000001 ($L_1 = 1$). The shortest distance among these values can be easily estimated by computing the bitwise-AND operation without performing value comparison:

$$00111111 \ \& \ 00001111 \ \& \ 00000001 = 00000001$$

Fig. 1 illustrates a 2D example where two binary kernels of size 5×5 with 4 b masks encoding are centered, respectively, on a red and a blue cell (Fig. 1(b)). The bit mask encodes the L_1

distance, from 0000 at the kernel center ($L_1 = 0$) to 1111 at the corners ($L_1 = 4$), as shown in Fig. 1(a). In Fig. 1(c), both kernels are applied using bitwise-AND around their respective centers. In the overlapping region (outlined in black), we can see how the method preserves the shortest distance to the nearest center. All map cells were initialized to 1111 (the truncation distance) before applying the binary kernels, indicating the maximum or unknown distance.

B. TDF Grid Map

Notice that the previous kernel actually represents the L_1 TDF of an obstacle in its center. We can easily compute the TDF of a general point cloud as a grid map by just performing the bitwise-AND of this kernel with the grid map at each cell that contains a point. The representation of the distances in the form of a bit-set mask implicitly solves all the problems related to the distance field computation. Thus, we do not need to know if one point is close to any other, or the order of evaluation of points in the cloud.

Another advantage of this approach is its easy parallelization. In principle, each point can update the grid map in parallel, and then merge all updates by means of bitwise-AND. We can make use of this feature to accelerate the computation of the TDF, distributing the computation among different threads in the CPU.

We store the TDF map with two levels of hashing: the first level splits space into 1 m^3 blocks, and the second level manages the high-resolution cells inside a block. Only blocks that actually receive points are expanded to high resolution (allocated on demand by the second level). Each allocated fine cell stores the L_1 distance to the closest point of the 3D map to that cell, or the truncation distance if the closest point is far. The truncation distance will be defined by three factors: the number of bits used to represent the bit-set mask, the size of the kernel, and the grid resolution. The number of bits of the mask must be equal to the maximum L_1 distance into the kernel in order to make the approach efficient, so the truncation distance will be finally given by the size of the kernel and the resolution.

C. Grid Map Initialization and Access

As previously introduced, we organize the map with a two-level (double) hashing scheme designed for memory efficiency. The first level partitions space into fixed 1 m^3 blocks. Many of these blocks will never receive measurements in typical scenes, so they remain lightweight: each block simply holds a pointer that is null until needed. When a point falls inside a block for the first time, that pointer is set to a newly allocated high-resolution container for that block. At the second level, the fractional part of the coordinates (scaled by the chosen resolution) determines the fine cell inside the block where the binary mask value is stored. This on-demand allocation ensures that only blocks receiving points are expanded to high resolution, while all other regions remain unallocated. If the maximum number of fine containers is reached, the oldest one is recycled through a circular buffer, keeping memory bounded and efficient.

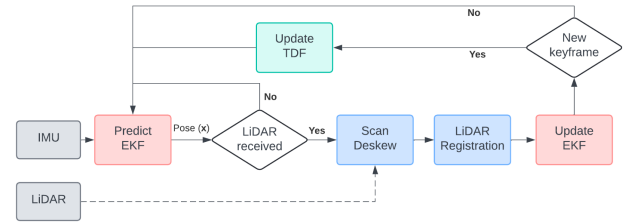


Fig. 2. D-LIO Workflow. Red indicates the Kalman filter, green the TDF grid map, and blue the LiDAR preprocessing and optimization.

To ensure correct operation of the binary kernel with bitwise-AND, high-resolution containers are initialized to the maximum truncation distance, that is, a mask with all bits set. This value will be updated by the binary kernel as soon as the value to store into the grid cell is smaller than the truncation distance. Containers that remain unused are not allocated, and any recycled container (when hitting the memory cap) is re-initialized to the maximum truncation distance before reassignment.

D. Distance Computation

The grid map stores binary masks, not distances. We need a method to compute the actual L_1 distance from the binary mask in the grid cells. We make use of the C++ `std::bitset()` standard function, which is directly mapped into computer assembly instructions. This function counts the number of bits set into a register, returning an integer with such number, the corresponding L_1 distance.

E. Distance Interpolation

Following [34], we make use of trilinear interpolation in order to estimate the distance field beyond the grid cell resolution. However, here, the interpolation is computed online.

IV. DIRECT 3D LiDAR-INERTIAL ODOMETRY

Fig. 2 shows a block diagram of the D-LIO approach. The algorithm incrementally estimates the system pose using an Extended Kalman Filter (EKF) by integrating inertial measurements for pose prediction and point cloud to map registration for pose update. Thus, for each incoming point cloud, the system performs a cloud registration by aligning it with the environment mapped so far based on the Truncated Distance Field (TDF), using the EKF estimate as initial guess. This alignment is used to constrain the EKF's pose and velocity by means of state update. Whenever the estimated pose variation exceeds a predefined threshold, the TDF map is updated by incorporating new information. This updated map then facilitates a more precise alignment of subsequent point clouds during the registration process.

A. Inertial State Integration

Following LiDAR odometry common practices [35][24], inertial sensors can be used to improve state estimation between scans. Inertial integration provides a good initial solution for

LiDAR scan alignment, which benefits the subsequent scan-matching process. In addition, high-frequency inertial integration enables LiDAR scan deskewing, which also positively impacts the quality of the scan-matching process. We make use of an EKF with the following state vector:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{a}_b, \mathbf{r}, \mathbf{g}_b]^t \quad (1)$$

where the state encapsulates the 3D position $\mathbf{p} = (x, y, z)$, linear velocity $\mathbf{v} = (v_x, v_y, v_z)$, accelerometer bias $\mathbf{a}_b = (a_{bx}, a_{by}, a_{bz})$, orientation $\mathbf{r} = (r_x, r_y, r_z)$ and gyroscope bias $\mathbf{g}_b = (g_{bx}, g_{by}, g_{bz})$. Here, the position \mathbf{p} , velocity \mathbf{v} , and orientation \mathbf{r} are expressed in the global frame, while the accelerometer and gyroscope biases, \mathbf{a}_b and \mathbf{g}_b respectively, are expressed in the local inertial frame.

In the prediction step, the filter integrates IMU measurements (angular velocity and linear acceleration) to propagate the state forward in time. Once a new LiDAR scan is received, deskewing is performed using the high-rate inertial state estimates to compensate for motion-induced distortions accumulated during the scan acquisition. Subsequently, the update step is carried out using the refined pose from the cloud registration step and the estimated velocity derived from successive point cloud positions.

B. LiDAR Point-Cloud Registration

This point cloud registration is formulated as a nonlinear optimization problem over the SE(3) space, where the objective is to minimize the distance from each point to the closest object in the map. The point-cloud is pre-transformed according to the EKF current state, so that the registration only needs to deal with the accumulated error in the inertial integration between scans.

The problem is formulated as the following minimization of a sum of residuals:

$$\min_{\mathbf{t}, \mathbf{q}} \sum_i \rho_i L_1(\mathbf{R}(\mathbf{q}) \cdot \mathbf{p}_i + \mathbf{t})^2 \quad (2)$$

where i corresponds to each point in the cloud, $\mathbf{t} \in \mathbb{R}^3$ is the translation vector, $\mathbf{q} \in \mathbb{H}$ is the unit quaternion encoding rotation, $\mathbf{R}(\mathbf{q})$ is the corresponding rotation matrix, ρ_i is the robust kernel detailed in Section IV-C, and $L_1(\cdot)$ returns the unsigned L_1 distance to the nearest object using trilinear interpolation over the TDF computed so far as explained in Sections III-D and III-E.

The optimization is carried out using the Ceres Solver library [36]. To properly handle the unit norm constraint of quaternions, a proper Lie group formulation is applied, along with its corresponding Lie algebra, enabling optimization in the tangent space.

C. Outlier Rejection

Before formulating the optimization problem, points that fall outside the TDF grid are discarded to ensure valid distance evaluations. For the remaining points, a robust loss function is applied to mitigate the effect of outliers while preserving global consistency in the estimation. In particular, a Cauchy loss is employed, whose scale parameter is a function of the point's

distance to the sensor:

$$\rho_i = \text{CauchyLoss}(\lambda \cdot (0.1 + 0.1 \cdot \|\mathbf{p}_i\|)) \quad (3)$$

where λ is a tuning factor and $\|\mathbf{p}_i\|$ denotes the Euclidean norm of the 3D point \mathbf{p}_i expressed in the LiDAR sensor system reference.

This robust kernel ensures that distant points are not excessively penalized due to their higher residuals, which can naturally arise from small orientation errors. For example, a rotation error of just 1° may produce negligible displacements in nearby points (e.g., a few centimeters), but can lead to significant deviations (e.g., a few meters) in distant points. If a fixed-scale robust loss were used, these far points could be interpreted as outliers and ignored during optimization, causing the system to converge to a locally optimal but globally inaccurate solution. The proposed kernel preserves the contribution of both near and far points, promoting robust convergence across the entire cloud.

D. Key-Framing and Map Updating

A keyframe-based approach is employed for map updates, where a new keyframe is created whenever the relative translation or rotation with respect to the previous keyframe exceeds the predefined thresholds t_{th} and q_{th} , respectively. When a new keyframe is created, the TDF map is updated by integrating the current point cloud, which is first transformed according to the latest optimized pose estimate, ensuring proper alignment in the global frame before fusion. This process, detailed in Section III, updates the TDF grid by performing bitwise-AND operations of the kernel centered at each point, integrating it from the transformed cloud into the grid while considering both the new point and the existing map data. Although it is the most computationally expensive stage of the system, the use of bitwise-AND operations and parallelization helps optimize the map update process and significantly reduce processing time.

As noted earlier, in case all high-resolution containers are already in use when new points arrive (e.g., as the vehicle enters previously unseen areas), the system removes the information from the oldest 1 m^3 block and reassigns its high-resolution container to the newly observed block in the current scan, keeping memory bounded while prioritizing currently visible regions.

V. EXPERIMENTAL RESULTS

The algorithm has been tested on three datasets, the **VIRAL Dataset** [37], the **Newer College Dataset** [38] and the **VBR Dataset** [39], to evaluate its ability to correctly localize in different environments using diverse trajectories that challenge its performance.

The accurate estimation of odometry has been primarily validated with the VIRAL Dataset, as it features more challenging environments and complex aerial trajectories that test the algorithm's stability. We evaluate the sequences "eee", "nya", and "sbs", using a DJI M600 Hexacopter equipped with two LiDARs (Ouster OS1-16 gen 1, horizontal and vertical), providing a combined density of approximately 20 k points/scan. For the Newer College Dataset ("Quad-Easy" trajectory), we

TABLE I

EXPERIMENTAL RESULTS ON VIRAL AND NEW COLLEGE DATASET: ATE (M) FOR ALL METHODS (M). THE BEST RESULT IS HIGHLIGHTED IN **BOLD** AND THE SECOND BEST IS UNDERLINED, EXCLUDING LIO-SAM AND PIN-SLAM AS THEY PERFORM BACK-END OPTIMIZATION. D-LIO, FAST-LIO2, AND LIO-SAM ARE LiDAR-INERTIAL METHODS, WHILE THE OTHERS RELY SOLELY ON LiDAR. METRICS WERE TAKEN FROM [40], [41] AND [42]

	D-LIO	A-LOAM	M-LOAM	KISS-ICP	FAST-LIO2	LIO-SAM	PIN-SLAM
eee01	0.074	0.212	0.249	2.383	<u>0.166</u>	0.075	-
eee02	0.073	0.199	0.168	1.586	<u>0.100</u>	0.069	2.056
eee03	0.121	0.148	0.233	1.055	<u>0.142</u>	0.101	0.615
nya01	<u>0.080</u>	0.077	0.123	0.359	0.127	0.076	0.086
nya02	<u>0.104</u>	0.091	0.191	-	0.151	0.07	0.092
nya03	0.065	<u>0.108</u>	0.226	1.389	0.130	0.137	0.455
sbs01	0.085	<u>0.103</u>	0.173	1.353	0.130	0.089	0.205
sbs02	0.067	<u>0.091</u>	0.147	1.435	0.144	0.083	0.528
sbs03	0.098	<u>0.367</u>	0.153	1.037	<u>0.126</u>	0.14	0.708
q-easy	<u>0.093</u>	0.085	0.141	0.100	0.100	0.074	0.09

TABLE II

EXPERIMENTAL RESULTS ON VBR DATASET: ATE (M) AND RPE (%) IN PARENTHESIS FOR ALL METHODS. THE BEST RESULT IS HIGHLIGHTED IN **BOLD** AND THE SECOND BEST IS UNDERLINED

	D-LIO	KISS-ICP	PIN-SLAM
diag	<u>0.560</u> (0.72)	1.397 (1.790)	0.362 (0.468)
pincio	7.39 (3.45)	<u>0.784</u> (0.485)	0.647 (0.453)

evaluate the reconstruction quality obtained from the TDF map, leveraging existing benchmarks on this dataset that report results across different approaches. Finally, the VBR dataset is used to demonstrate the algorithm’s scalability in a significantly larger environment, showcasing its ability to handle increasing map size and dense point clouds. In this case, we evaluate two trajectories: “diag”, an indoor/outdoor sequence of 1.4 km across three building floors with several loops, and “pincio”, an outdoor 2.5 km trajectory covering a wide open area with dense vegetation. In both datasets, the platform is handheld with an Ouster OS-0-128 LiDAR; with 1024 columns for the Newer College (~130 k points/scan) and 2048 columns for the VBR (~262 k points/scan).

Our results are presented in Tables I and II together with other approaches, such as A-LOAM [18], M-LOAM [19], FAST-LIO2 [24], KISS-ICP [2], LIO-SAM [1] and PIN-SLAM [27]. In all three datasets, the Absolute Translation Error (ATE) was computed using the benchmarking tools provided by the VIRAL dataset to ensure a consistent and reliable trajectory evaluation. For the VBR dataset, we also report the Relative Pose Error (RPE), as it is especially informative for long trajectories. The plots of all trajectories, together with complementary information and 3D reconstructions, are available in the supplementary video¹ and repository.² The experiments were performed on an HP Victus 16 laptop equipped with 32 GB RAM and a 13th generation Intel Core i7-13700H processor. No GPU was used for the experimentation.

A. Trajectory Evaluation

These experimental results made use of Fast-TDF using 64 b mask encoding together with a grid resolution of 0.05 m, the

¹ <https://www.youtube.com/watch?v=HmrA9YOCZ9w>

² <https://github.com/robotics-upo/D-LIO.git>

truncation distance of this kernel is approximately 3 meters. The overall map extent is dataset-dependent; however, by using the proposed coarse layer of 1 m³ blocks, we can scale the map footprint to very large areas at low resolution while allocating high-resolution cells only where points are observed. In our largest runs, we scaled the map to 500 × 500 × 100m. Finally, the rotation/translation thresholds used to update the grid were set to $\lambda = 2.0$, $t_{th} = 1$ m, and $q_{th} = \pi$ rad. As shown in Table I, our method attains the lowest ATE in most sequences and ranks second-best in the remainder, highlighting its consistent accuracy and robustness across diverse and challenging trajectories.

Please, notice that LIO-SAM and PIN-SLAM performs back-end optimization, which makes the comparison with pure odometry not entirely fair. However, we have included its results to demonstrate that our approach, despite not using back-end optimization, achieves comparable performance.

B. Scalability

After validating the accuracy of our algorithm in more dynamic trajectories, we next assess scalability on longer trajectories and larger maps using the VBR Dataset (See Table II). On the “diag” sequence, we allocated 250 k high-resolution blocks within a 150 × 150 × 50 m map. Despite the moderate map size, the trajectory is long (1.4 km); however, the points are compactly distributed, so this allocation sufficed to represent the space without reallocations, as cells are created only around observed points, avoiding empty space and improving memory efficiency. Results are competitive with the state of the art, second only to PIN-SLAM, which benefits from global loop closures on long, loopy paths. In the “pincio” sequence, the platform covers 2.5 km over a much wider area. Since the full map cannot be kept in memory without reallocations, we limited the allocation to 50 k high-resolution blocks. The system preserves detail only near the current pose, dynamically evicting distant blocks that no longer affect the current scan, thus managing memory dynamically. As this run spans a larger area and lacks loop closures, the absolute error increases; nevertheless, an ATE of 7 m over 2.5 km corresponds to only 0.28% of the total trajectory length.

C. Mapping

To quantitatively evaluate the mapping performance of our algorithm, we adopt standard metrics widely used for 3D reconstruction benchmarking, following the evaluation methodology proposed in [43]. The evaluation includes four key indicators: Chamfer-L1 distance (C-L1), reconstruction accuracy, completeness, and F-Score. Chamfer-L1 offers an aggregate measure integrating both reconstruction accuracy, representing the fidelity to the ground truth, and completeness, which reflects how thoroughly the environment has been captured. For accuracy, completeness, and Chamfer-L1 metrics, smaller values represent improved map quality. Conversely, higher F-Score values reflect a better balance between precision and recall. The detailed results of these metrics are summarized in Table III

Overall, D-LIO remains at the state of the art: it achieves the best completeness (10.63), second-best Chamfer-L1 (11.39)

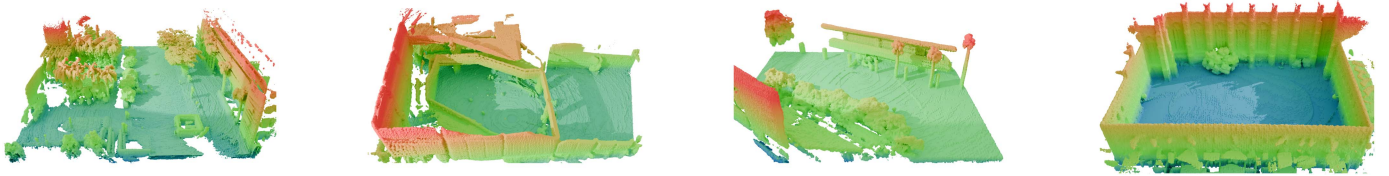


Fig. 3. 3D map reconstructions (from left to right) of “eee”, “nya”, “sbs”, and Newer College (Quad-Easy) datasets.

TABLE III
 MAPPING METRICS ON NEWER COLLEGE QUAD-EASY SEQUENCE. BEST AND SECOND BEST SCORES ARE MARKED IN **BOLD** AND UNDERLINED RESPECTIVELY

	Pose	Acc.	Comp.	C-L1	F-Score
VDB-Fusion	KissICP	14.03	25.46	19.75	69.50
SHINE		14.87	20.02	17.45	68.85
NKSR		15.67	36.87	26.67	58.57
PUMA	Odometry	15.30	71.91	43.60	57.27
SLAMesh		19.21	48.83	34.02	45.24
NeRF-LOAM		12.89	22.21	17.55	74.37
S²KAN-SLAM		13.32	18.80	16.06	72.03
ImMesh		15.05	19.80	17.42	66.87
PIN-SLAM		<u>11.55</u>	15.25	13.40	82.08
KN-LIO		8.18	<u>11.65</u>	9.92	91.01
D-LIO		12.14	10.63	<u>11.39</u>	83.83

and F-Score (83.83), and competitive accuracy (12.14). It approaches KN-LIO’s top performance and surpasses PIN-SLAM on Chamfer-L1, completeness, and F-Score, while classical TSDF/SDF and several odometry-driven baselines lag behind.

D. Runtime and Memory Consumption

The performance of the system was analyzed in terms of memory usage, CPU load, and execution time for all datasets. Overall memory consumption is governed by the number of high-resolution blocks (each a 1 m³ region discretized at 0.05 m) rather than the coarse map, whose footprint is negligible by comparison (see Sec. V-A). In VIRAL (largest trajectory), we used 180 k blocks (31% of RAM). In Newer College, due to the higher point concentration, only 45 k blocks (22% of RAM) were required. On *diag*, despite a 1.4 km path within one building, most points fell into already-activated regions; with an intentionally oversized budget of 250 k blocks (47% of RAM) we needed no reallocations. Finally, on *pincio*, points were widely dispersed with little coarse-block reuse, so we ran a smaller allocation of 45 k blocks (22% of RAM) with dynamic reallocation as the platform advanced.

Execution times are summarized in Table IV. The map update step normally emerges as the most computationally expensive stage, as its time strongly depends on the point cloud density. For instance, in the VIRAL dataset, the lower density from OS-1 sensors (≈ 20 k points/scan) leads to shorter update times, whereas in the college and VBR dataset, the OS-0 sensor produces ≈ 100 -260 k points/scan making updates slower. However this cost also depends on environment topology: in *diag*, compact distributions and frequent revisits mean many already-seen regions

TABLE IV
 RUNTIME ANALYSIS ACROSS DIFFERENT DATASETS WITH NO DOWNSAMPLE, SHOWING MEAN \pm STANDARD DEVIATION (IN SECONDS) FOR EACH PROCESSING PHASE. ALSO REPORTED, PER TRAJECTORY, THE AVERAGE COUNT OF TRULY MEASURED LiDAR RETURNS

	Points	Total (s)	Optimize (s)	Update (s)
eee	20k	0.048 \pm 0.055	0.046 \pm 0.013	0.155 \pm 0.009
nya	20k	0.049 \pm 0.034	0.042 \pm 0.014	0.146 \pm 0.016
sbs	20k	0.051 \pm 0.048	0.044 \pm 0.016	0.142 \pm 0.034
q-easy	100k	0.479 \pm 0.196	0.442 \pm 0.158	0.470 \pm 0.095
diag	190k	0.603 \pm 0.240	0.565 \pm 0.194	0.315 \pm 0.085
pincio	140k	0.576 \pm 0.256	0.505 \pm 0.182	0.563 \pm 0.146

are not updated, so the update time can match, or even be lower than, the optimization.

Additionally, updates are triggered at a much lower frequency than optimization (which runs at the sensor rate, 10 Hz), so the weighted average tends to be driven by the optimization step, also influenced by the point density. Note that all results were obtained under a worst-case computational load, no downsampling; in practice, applying voxel filtering can significantly speed up the system with negligible impact on accuracy.

It is also worth noting that state-of-the-art distance field mapping approaches on CPU, such as VDB-GPDF [13], which focuses solely on reconstruction, reported similar execution times to ours, but our approach performs both odometry and reconstruction without any downsampling. Although the map is updated less frequently, once every meter, it integrates the full-resolution point cloud, offering a balanced trade-off between computational efficiency and mapping accuracy.

VI. CONCLUSIONS AND FUTURE WORK

D-LIO, a fully direct LiDAR–inertial odometry pipeline based on Fast-TDF simultaneous mapping, has been shown to match or outperform feature-driven state-of-the-art methods while producing a continuously available distance field for downstream tasks. Unlike conventional Neural TSDF/ESDF frameworks whose update cost grows with map volume, our Fast-TDF binary-mask scheme ties each integration step solely to the number of incoming points. By encoding obstacle proximity directly in the truncated distance field, we avoid feature extraction, yielding robust convergence even in feature-sparse or highly reflective environments.

Future work will focus on extending the Fast-TDF kernel to support signed distance estimation, enabling the system to distinguish whether a point lies in front of or behind an obstacle, potentially enhancing registration accuracy.

REFERENCES

- [1] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.
- [2] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In defense of point-to-point ICP simple, accurate, and robust registration if done the right way," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.
- [3] J. Zhang and S. Singh, "LOAM : LiDAR odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst. Conf.*, 2014, pp. 109–111.
- [4] K. Chen, B. T. Lopez, A.-A. Agha-mohammadi, and A. Mehta, "Direct LiDAR odometry: Fast localization with dense point clouds," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2000–2007, Apr. 2022.
- [5] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.
- [6] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.
- [7] A. Segal, D. Hähnel, and S. Thrun, "Generalized-ICP," 06 2009.
- [8] Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro, C. Stachniss, and M. Chli, "Voxfield: Non-projective signed distance fields for online planning and 3D reconstruction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 5331–5338.
- [9] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM Trans. Graph.*, vol. 32, pp. 27:1–27:22, 2013.
- [10] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn. - SIGGRAPH '96*, 1996, pp. 303–312.
- [11] J. Ortizet al., "iSDF: Real-time neural signed distance fields for robot perception," 2022. [Online]. Available: <https://arxiv.org/abs/2204.02296>
- [12] V. Vasilopoulos, S. Garg, J. Huh, B. Lee, and V. Isler, "HIO-SDF: Hierarchical incremental online signed distance fields," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 17537–17543.
- [13] L. Wu, C. L. Gentil, and T. Vidal-Calleja, "VDB-GPDF: Online Gaussian process distance field with VDB structure," *IEEE Robot. Autom. Lett.*, vol. 10, no. 1, pp. 374–381, Jan. 2025.
- [14] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
- [15] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, vol. 3, pp. 2743–2748.
- [16] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time elastic LiDAR odometry with loop closure," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 5580–5586.
- [17] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct LiDAR-inertial odometry: Lightweight LIO with continuous-time motion correction," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3983–3989.
- [18] T. Wang, X. Wu, H. Yang, P. Zhao, and Q. Hu, "An improved a-loam 3D mapping method based on ultra-wide field angle of multi-line LiDAR," in *Proc. 29th Int. Conf. Geoinformatics*, 2022, pp. 1–6.
- [19] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust odometry and mapping for multi-LiDAR systems with online extrinsic calibration," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 351–371, Feb. 2022.
- [20] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMA++ : Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.
- [21] Q. Li et al., "LO-Net: Deep real-time LiDAR odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8465–8474.
- [22] C. Zheng, Y. Lyu, M. Li, and Z. Zhang, "LodoNet: A deep neural network with 2D keypoint matching for 3D LiDAR odometry estimation," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 2391–2399, doi: [10.1145/3394171.3413771](https://doi.org/10.1145/3394171.3413771).
- [23] Y. Cho, G. Kim, and A. Kim, "Unsupervised geometry-aware deep LiDAR odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2145–2152.
- [24] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.
- [25] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4861–4868, Apr. 2022.
- [26] J. Deng et al., "NeRF-LOAM: Neural implicit representation for large-scale incremental LiDAR odometry and mapping," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8218–8227.
- [27] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency," *IEEE Trans. Robot.*, vol. 40, pp. 4045–4064, 2024, doi: [10.1109/TRO.2024.3422055](https://doi.org/10.1109/TRO.2024.3422055).
- [28] Z. Wang, L. Ren, Y. Wen, and H. Wang, "KN-LIO: Geometric kinematics and neural field coupled LiDAR-inertial odometry," 2025. [Online]. Available: <https://arxiv.org/abs/2501.04263>
- [29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 1366–1373, doi: [10.1109/IROS.2017.8202315](https://doi.org/10.1109/IROS.2017.8202315).
- [30] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: Flexible and efficient TSDF integration of range sensor data," *Sensors*, vol. 22, 2022, Art. no. 1296.
- [31] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.
- [32] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, "Shine-mapping: Large-scale 3D mapping using sparse hierarchical implicit neural representations," 05 2023, pp. 8371–8377.
- [33] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 3437–3444.
- [34] F. Caballero and L. Merino, "DLL: Direct LiDAR localization. a map-based localization approach for aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5491–5498.
- [35] Y. Wu, T. Guadagnino, L. Wiesmann, L. Klingbeil, C. Stachniss, and H. Kuhlmann, "LIO-EKF: High frequency LiDAR-inertial odometry using extended Kalman filters," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 13741–13747.
- [36] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres solver," 2023. [Online]. Available: <https://github.com/ceres-solver/ceres-solver>
- [37] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "NTU viral: A visual-inertial-ranging-LiDAR dataset, from an aerial vehicle viewpoint," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 270–280, 2022.
- [38] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera LiDAR inertial extension to the newer college dataset," 2022. [Online]. Available: <https://arxiv.org/abs/2112.08854>
- [39] L. Briziet al., "VBR: A vision benchmark in rome," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 15 868–15 874.
- [40] T.-M. Nguyen, S. Yuan, M. Cao, T. H. Nguyen, and L. Xie, "Viral slam: Tightly coupled camera-IMU-UWB-LiDAR SLAM," 2021. [Online]. Available: <https://arxiv.org/abs/2105.03296>
- [41] J. L. Blanco-Claraco, "A flexible framework for accurate LiDAR odometry, map manipulation, and localization," in *Proc. Int. J. Robot. Res.*, vol. 44, no. 9, pp. 1553–1599, 2025. [Online]. Available: <https://doi.org/10.1177/02783649251316881>
- [42] D. Chung and J. Kim, "NV-LIOM: LiDAR-inertial odometry and mapping using normal vectors towards robust SLAM in multifloor environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 9375–9382, Nov. 2024.
- [43] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Jul. 2017, doi: [10.1145/3072959.3073599](https://doi.org/10.1145/3072959.3073599).