2FAST-2LAMAA: A Lidar-Inertial Localisation and Mapping Framework for Non-Static Environments

Cedric Le Gentil[1], Raphael Falque[1] and Teresa Vidal-Calleja[1]

Abstract— **This document presents a framework for lidar-inertial localisation and mapping named 2Fast-2Lamaa. The method revolves around two main steps which are the inertial-aided undistortion of the lidar data and the scan-to-map registration using a distance-field representation of the environment. The initialisation-free undistortion uses inertial data to constrain the continuous trajectory of the sensor during the lidar scan. The eleven DoFs that fully characterise the trajectory are estimated by minimising lidar point-to-line and point-to-plane distances in a non-linear least-square formulation. The registration uses a map that provides a distance field for the environment based on Gaussian Process regression. The pose of an undistorted lidar scan is optimised to minimise the distance field queries of its points with respect to the map. After registration, the new geometric information is efficiently integrated into the map. The soundness of 2Fast-2Lamaa is demonstrated over several datasets (qualitative evaluation only). The real-time implementation is made publicly available at https://github.com/UTS-RI/2fast2lamaa.**

I. INTRODUCTION

Thanks to a high level of accuracy and an ever-decreasing cost, lidars have become omnipresent in the robotics field. This gain in popularity is explained by the reliability of the provided geometric measurements and their low noise level over large distances (1-200m). While technology is evolving, nowadays lidars still consist of a limited amount of laser beams that sweep the environment. This sweeping action implies that lidar scans are subject to motion distortion when the sensor moves. A common approach to correct the distortion is using information from the accelerometer and gyroscope of an Inertial Measurement Unit (IMU). Given a set of initial conditions, the inertial data can be used to predict the sensor position and thus correct the motion distortion of the lidar data. However, in many scenarios, the initial conditions are not readily available. This document presents a framework for lidar-inertial localisation and mapping in non-static environments that addresses the motion distortion issue without the need for any initialisation procedure following concepts from our previous works [1] and [2]. We have named it Fast Field-based Agent-Subtracted Tightly-coupled Lidar Localisation And Mapping with Accelerometer and Angular-rate (2Fast-2Lamaa).

A key aspect of 2Fast-2Lamaa is that it can be operated in dynamic environments with people moving in the sensor's surroundings and still provide a clean "people-free" map of the environment. This feature is particularly useful in applications where a map of the environment is created once

[1]All authors are with the Robotics Institue at the University of Technology Sydney: cedric.legentil@uts.edu.au

(a) Point cloud map   (b) Mesh map

Fig. 1.   Example of the mapping result using 2FAST-2LAMAA on the Newer College dataset [3]. (a) is the point cloud map in blue/grey, the current lidar scan in tile teal, and the distance function around the sensor with a jet colourmap. (b) is the final mesh reconstruction.

and then reused for localisation only. It alleviates the need for manual removal of trails in the map left by the presence of dynamic objects during the mapping stage. Concretely, 2Fast-2Lamaa builds atop some previous works [4]–[6]. It performs optimisation-based lidar-inertial undistortion, filters dynamic and unreliable points from the lidar scans, estimates the global pose with scan-to-map registration and map cleaning, and provides a triangle mesh of the environment via surface reconstruction (c.f. Fig. 1).

The proposed map representation is an efficient approximation of our Gaussian Process (GP)-based distance field from [5]. Accordingly, 2Fast-2Lamaa allows for fast queries of the shortest point-to-obstacle distance and the corresponding direction at any location in space. It is particularly useful for downstream applications such as path-planning (e.g., [7], [8]). While building on previous work, this work presents original contributions:

- A novel method for reliability score computation for fast scan-filtering (removing dynamic and unreliable points from lidar scans).
- Real-time scan-to-map registration using distance fields.
- A novel uncertainty proxy for the distance fields.
- The integration of an efficient map cleaning mechanism (a.k.a free-space carving) to clean the proposed map representation from objects no longer in the environment.
- The integration of screen Poisson Surface reconstruction [9] for triangle-mesh map generation.
- The open-source implementation of all the aforementioned components.

In its current form, this document mainly focuses on the

Fig. 2. Method overview.

technical aspects of the framework to make it available to the research community. Accordingly, there is no dedicated related work section and the experimental analysis is limited to qualitative results. Future work will widen the scope of this document. Meanwhile, for an overview of the related work we invite the reader to read [4]–[6].

II. FRAMEWORK OVERVIEW

Let us consider a rigidly mounted 6-DoF IMU and lidar with $\mathbf{T}_I^L$ the homogeneous geometric transformation between the two sensors. We aim at estimating lidar pose $\mathbf{T}_W^{L_t}$ in a world frame $\mathfrak{F}_W$ through time in dynamic environments as well as establishing a map of the environment. The IMU consists of a 3-axis gyroscope and a 3-axis accelerometer and its pose $\mathbf{T}_W^{I_t}$ is linked to the lidar pose as $\mathbf{T}_W^{I_t} = \mathbf{T}_W^{L_t} \mathbf{T}_I^{L^{-1}}$.

As shown in Fig. 2, the proposed framework first performs local undistortion of the lidar measurements by tightly

coupling the geometric and inertial data as in [6]. The core principle is the use of continuous IMU preintegration [10] as the continuous representation of the 6-DoF trajectory of the sensor suite. Associated with point-to-line and point-to-plane lidar residuals in a non-linear least-square optimisation, the trajectory is estimated over short fixed periods of time.

Then, the undistorted geometric data is filtered to remove *unreliable* points from each point cloud. The definition of unreliable point includes points that belong to dynamic objects as well as points that do not provide much information about the underlying surface (ambiguity on the surface normal). This process is inspired by [4] as we locally analyse the spatiotemporal covariance of the point cloud to classify each point. The details of this novel filtering step are provided in Section III-B.

Clouds of reliable points are used to both perform scan-to-map registration of the data and increment a global map of the environment. The voxelised map representation is associated with accurate GP distance fields [5]. This allows for scan-to-map registration in a simple non-linear least-square formulation, and for downstream applications such as shape reconstruction or path planning. When including registered data into the global map, we introduce a *free space carving* step to remove points belonging to objects that are no longer present in the environment. An example is the case of cars parked on the side of the road: while being considered as static when first observed, the map should be updated when the place is revisited in the absence of the vehicles.

III. UNDISTORTION / ODOMETRY

A. Lidar-inertial undistortion

In this subsection, we provide a succinct summary of the lidar-inertial undistortion step of the proposed framework. We invite the reader to refer to our previous work in [6] for more details.

A key concept of the proposed lidar undistortion method is the redefinition of what is a lidar scan. Traditionally, a scan corresponds to the set of points collected by the lidar throughout the duration of a single sweeping pattern, typically a single $360°$ rotation for spinning lidars, and is considered as one measurement. Here we consider the lidar points independently and define a scan as the points collected between arbitrarily chosen timestamps τ_i and τ_{i+1}. Each of these scans undergoes a feature detection step where each point is given a roughness score that represents the level of planarity of the underlying surface. Points with low scores are selected as planar features, and the ones with high scores are used as edge features.

Using the continuous preintegration method introduced in [10], we can characterise the trajectory of the sensors continuously between τ_i and τ_{i+1} as

$$\mathbf{R}_{I_{\tau_i}}^t = \Delta \mathbf{R}_{\tau_i}^t \tag{1}$$

$$\mathbf{p}_{I_{\tau_i}}^t = (t - \tau_i)\mathbf{v}_{I_{\tau_i}}^{\tau_i} + \frac{(t - \tau_i)^2}{2}\mathbf{g}_{I_{\tau_i}} + \Delta \mathbf{p}_{\tau_i}^t, \tag{2}$$

with $\mathbf{R}_{I_{\tau_i}}^t$ and $\mathbf{p}_{I_{\tau_i}}^t$ the rotational and translational components of $\mathbf{T}_{I_{\tau_i}}^t$, $\Delta \mathbf{R}_{\tau_i}^t$ and $\Delta \mathbf{p}_{\tau_i}^t$ the preintegrated measurements, $\mathbf{v}_{I_{\tau_i}}^{\tau_i}$ the initial velocity of the IMU, and $\mathbf{g}_{I_{\tau_i}}$ the gravity vector in the IMU frame at time τ_i. While omitted for the sake of notation clarity, the preintegrated measurements are functions of the gyroscope and accelerometer biases [10]. Assuming constant biases between τ_i and τ_{i+1} and linearising the preintegrated measurements, the trajectory is fully characterised by eleven Degree-of-Freedoms (DoFs) with three for $\mathbf{v}_{I_{\tau_i}}^{\tau_i}$, two for the direction of $\mathbf{g}_{I_{\tau_i}}$, and six for the biases.

By associating together the aforementioned planar and edge feature points across consecutive scans using KDTrees [11], and by using the preintegration-based continuous trajectory (1) and (2), the scans are unistorted through the minimisation of point-to-plane and point-to-line distances in a non-linear least-square formulation. The eleven DoFs of the state are estimated using the Levenberg-Maquardt algorithm.

B. Point filtering

This step is inspired by our previous work on dynamic object detection in [4] which allowed us to individually label points in lidar scans as dynamic or not. Here we relax the classification problem into *reliable* and *unreliable* points. The set of unreliable points includes points from dynamic objects as well a points that are deemed to be untrustworthy in their ability to represent the local shape of the environment. In other words, an unreliable point and its spatial neighbourhood do not allow for robust estimation of the underlying surface normal vector. Similarly to [4], the core principle is to analyse the spatiotemporal patterns of each lidar/depth-camera point and their spatial neighbourhood. While [4] analyses the spatio-temporal normal vector computed with the Principal Component Analysis (PCA) of the local spatio-temporal covariance matrix, here we directly process the covariance matrix. Avoiding the Eigendecomposition of PCA allows for significantly faster computations.

Provided with a sequence of undistorted scans, the points are sorted in voxels of predefined size. For each voxel, the spatio-temporal covariance matrix $\mathbf{Q}_i$ associated to a point $\mathbf{x}_i$ is computed as:

$$\mathbf{Q}_i = \frac{1}{|\mathfrak{N}_i|}\left(\sum_{j \in \mathfrak{N}_i} \hat{\mathbf{x}}_j \hat{\mathbf{x}}_j^\top\right) - \mathbf{m}_i \mathbf{m}_i^\top, \qquad (3)$$

with

$$\mathbf{m}_i = \frac{1}{|\mathfrak{N}_i|}\sum_{j \in \mathfrak{N}_i} \hat{\mathbf{x}}_j, \qquad (4)$$

$\hat{\mathbf{x}}_i$ the time-augmented lidar points as $\begin{bmatrix} \mathbf{x}_i & t_i \end{bmatrix}^\top$, and $\mathfrak{N}_i$ the set of lidar points in the spatial neighbourhood of $\mathbf{x}_i$. Checking the correlation between the the temporal component and the spatial ones provides an efficient way to classify the points belonging to the voxel. Concretely, with the correlation matrix $\mathbf{C}_i$ as

$$\mathbf{C}_i = \mathrm{diag}(\mathbf{Q}_i)^{-\frac{1}{2}} \mathbf{Q}_i \mathrm{diag}(\mathbf{Q}_i)^{-\frac{1}{2}} = \begin{bmatrix} 1 & c_{xy} & c_{xz} & c_{xt} \\ c_{xy} & 1 & c_{yz} & c_{yt} \\ c_{xz} & c_{yz} & 1 & c_{zt} \\ c_{xt} & c_{yt} & c_{zt} & 1 \end{bmatrix} \qquad (5)$$

Fig. 3. Example of high spatiotemporal correlation in lidar data while the local neighbourhood reliably represents the underlying surface. The example is based on a point cloud that consists of three consecutive scans collected at three different poses and timestamps as highlighted by the red, green, and blue colours.

the points in the voxels are considered reliable if

$$\sqrt{c_{xt}^2 + c_{yt}^2 + c_{zt}^2} < thr. \qquad (6)$$

Additional checks are implemented to account for very specific scenarios as shown in Fig. 3. Due to the sparse nature of nowadays rotating lidars, the space-time correlation can be non-null in a point's neighbourhood while the corresponding points still provide trustworthy information of the true shape of the underlying surface. Formally, if (6) is not true, the spatial PCA is performed to check if the points lie on a near-perfect plane. If they do, the points in the voxel are considered to be reliable.

IV. MAPPING

The proposed mapping process is based on the GP-based distance field introduced in [5]. In simple words, [5] perform standard GP regression considering the 3D points of a point-cloud map as observations of a latent space that is equal to one on the environment's surface and fading to zero away from it. This latent space can be interpreted as a 3D occupancy field/function. Applying a particular function over the GP-inferred latent space allows for the accurate estimation of the distance to the closest element of the map. It has been demonstrated that such a method provides good non-parametric interpolation capabilities enabling the use of relatively sparse input point clouds. While providing high levels of accuracy, the original formulation of [5] suffers from the cubic $O(n^3)$ computational complexity of standard GP regression (n the number of points in the map).

In this work, we focus on approximating the distance field by only computing the GP inference locally leveraging efficient data structures to access a point and its spatial neighbourhood. While subdividing a large problem into smaller problems is a common strategy, it is important to not make parts of the map completely independent from one another. This would lead to the loss of interpolation abilities between each subdivision, thus hindering the accuracy and robustness of [5]. Accordingly, it is important that each local neighbourhood used for local inference possesses an overlap with surrounding neighbourhoods.

A. Map data structure

The key of the mapping approach is to store in parallel a sparse voxelised representation of the map and an associated spatial index $\mathfrak{S}$ that allows for closest neighbour and radius

Fig. 4. Example of the typical local neighbourhood used for GP inference. The green points are the centroids of the cells neighbouring the red cell.

(a) Without weighting   (b) With proposed weighting

Fig. 5. Illustration of the impact of the proposed weighting mechanism for our efficient GP-based distance field inference with sparse voxelised observations. The cell centroids are in red and the colourmap represents the distance field (the colour is purposely saturated at 1m for the sake of visibility. (a) is the inference with equal weight for each voxel observation. (b) is the inference with the proposed weighting based on the number of lidar point that occurred in each cell. One can clearly see the improvement in terms of smoothness, thus accuracy, of the field.

queries.[1] The voxelised representation consists of a hashmap mapping 3-integer tuples to *cells* and their various attributes. Among the cells' attributes, one can find the centroid of the points that occurred in the cell and the average direction vector from which the cell has been observed by the sensor.

Integrating new points in the map simply consists in checking if the corresponding cell is already present in the hashmap. If yes, the cell's parameters are incremented with the new point location and sensor position. Otherwise, a new cell is created and its position is added to the spatial index. Thanks to the hashmap properties, incrementing an existing cell is constant-time $\mathcal{O}(1)$ on average. The insertion of a new cell is slower with a $\mathcal{O}(log(n))$ complexity. Fortunately, this does not occur often (the map shown in Fig. 1 induced around 685k cell creations while more than 300 millions of lidar points have being processed).

B. Distance field query

The original formulation in [5] defines a latent space $o(\mathbf{x})$ that is arbitrarily set to one on the environment's surface and decreases toward zero further from the surface. The point cloud map $\mathbf{X}$ represents noisy unit measurements of the surface with σ the associated Gaussian uncertainty. The latent space is modelled with a GP as

$$o(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{7}$$

with k the covariance kernel function that specifies the covariance between two instantiations of $o(\mathbf{x})$. The latent field is inferred at any location following

$$\hat{o}(\mathbf{x}) = \mathbf{k}(\mathbf{x}, \mathbf{X})\left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma\mathbf{I}\right)^{-1}\mathbf{1}, \tag{8}$$

with $\mathbf{k}(\mathbf{x}, \mathbf{X})$ the covariance vector between the inference point and the observations, and $\mathbf{K}(\mathbf{X}, \mathbf{X})$ the covariance matrix of the observations. By applying a *reverting* function r to the latent field,[2] we recover the distance to the closest surface in the map as $d(\mathbf{x}) = r(o(\mathbf{x}))$. The main drawback

---

[1]One can think about the spatial index as being a standard KD-Tree [11] however our implementation is based on a different data structures (cf. Section V).

[2]We refer the reader to [5] for the exact expression of r.

of this approach is the computation complexity of the matrix in (8) as all the points in the point cloud map are considered.

In this work we exploit the sparse data structures discussed in Section IV-A to only perform (8) using a local neighbourhood of map points (as illustrated in Fig. 4), thus guaranteeing constant-time inference. Given a point $\mathbf{x}$ anywhere in space, we can query the closest cell in the voxelised map using the spatial index $\mathfrak{S}$, and then query the local neighbourhood of the voxel centroid. Doing so, it is possible to perform the distance field estimation $d(\mathbf{x})$ as in [5] locally resulting in a $\mathcal{O}(log(n))$ computational complexity due to the spatial index queries.

Directly using the cell centroids of the voxelised map as surface observation allows for extremely fast inference. Unfortunately, it impedes the probabilistic properties of the GP inference. Simply put, (8) give the same importance to each observation as per the measurement noise model is set to $\sigma\mathbf{I}$. Unfortunately, this is not realistic for voxel maps where one cell might correspond to 100 points while the neighbouring cell only to a couple of points. To alleviate this issue, we propose to "weight" the cell centroid observations based on the number of points that occurred in each cell. We call that number of points the *counter* c_i of a cell. Accordingly, in (8), we replace the measurement uncertainty model $\sigma\mathbf{I}$ with diag($\mathbf{w}$) where w_i (components of $\mathbf{w}$) are computed as a function of the cell's counter and the maximum cell counter in the neighbourhood. Note that a lower w_i means that the cell can be trusted more than a cell with a higher w_i. Thus the function that transforms c_i to w_i has to be decreasing. We picked a decreasing sigmoid function in the shape of $\frac{1}{1+\exp(-c_i/c_{\max})}$. Fig. 5 provides an illustration of the impact of the proposed weighting mechanism on the field's smoothness.

C. Distance field uncertainty proxy

Before starting this paragraph it is important to note that the distance field from our previous work [5] is not a GP as it is obtained via non-linear operations over a GP latent space. Accordingly, the naive GP variance inference [12] does not provide satisfying uncertainty information about the distance

values $d(\mathbf{x})$ away from the surface. Here, after reviewing the mechanisms introduced [5], we present a novel uncertainty proxy for the distance field.

To account for the uncertainty of the input point cloud, [5] introduced a method for optimising the observation noise σ based on a typical data sample. While this is a principled way to address the issue of noisy input, it presents limitations. One is the uniqueness of the noise parameter σ hindering accuracy when considering non-stationary noise (both in time and space). In realistic lidar-based odometry scenarios, some surfaces will appear "thicker" than others in the incoming point cloud due to various factors such as the noise in the lidar pose estimate, the angle of incidence of the lidar beams, etc. Regarding the uncertainty of the distance estimates, [5] provides an elegant uncertainty proxy based on the degree of similarity between the field's gradient and the expected gradient. However, this mostly addresses the issue of over-interpolation in tight spaces (corridor-like pattern with width in the order of magnitude of the kernel's lengthscale). We illustrate the aforementioned drawbacks in Fig. 6(c). Note that the proxy from [5] is not relevant to the proposed efficient distance field as the GP are computed locally considering a small neighbourhood, thus, rarely presenting corridor-like patterns.

Conceptually, the use of GP regression for point-cloud-based distance field estimation does not fit the original purpose of standard GP regression: with [5], we extrapolate the field far from the data observations whereas GP regression is an interpolation tool. In other words, we are seeking information and its variance in areas of space far from where the information is actually observed.[3] Here, the proposed distance uncertainty proxy "fetches" the information where it is relevant for a given distance query by first selecting the closest surface point to the query location (denoted $\mathbf{s}(\mathbf{x})$ in the following). The location of $\mathbf{s}(\mathbf{x})$ can easily be deduced from the inferred distance and gradient $\mathbf{s}(\mathbf{x}) = \mathbf{x} - d(\mathbf{x})\nabla d(\mathbf{x})$. The following step consists of comparing the local shape of the latent field o around $\mathbf{s}(\mathbf{x})$ with the one of a noiseless surface observation. We propose to use a local integration of the latent field to summarise the local shape of the latent field $\nu(\mathbf{s}(\mathbf{x})) = \iiint_{\mathbf{u} \in \mathcal{S}_{\mathbf{s}(\mathbf{x})}} o(\mathbf{u})d\mathbf{u}$, with $\mathcal{S}_{\mathbf{s}(\mathbf{x})}$ a small sphere around $\mathbf{s}(\mathbf{x})$. The integrals can be obtained with linear operators on the GP inference. Eventually, the proxy is defined as $\phi(\mathbf{x}) = |\nu(\mathbf{s}(\mathbf{x})) - \nu_{calib}|$, where ν_{calib} is computed with noiseless simulated data of a flat wall. It is equal to zero when the distance prediction is trustworthy and grows when the local shape differs from the ideal noiseless wall data.

The proposed proxy can be computed with the standard global GP from [5], but can also be made efficient with the sparse structure used in this work. First, the computation of $\mathbf{s}(\mathbf{x})$ can be omitted and approximated with the results of the spatial index query that already occurred to find the

---

³[3]Please note that the GP-inferred variance is still relevant close to the surface and can directly be leveraged in applications such as surface reconstruction.

(a) Inferred distance [m] (b) Distance field error [m]

(c) Uncertainty proxy from [5] (d) Novel uncertainty proxy

Fig. 6. Visualisation of the proposed uncertainty proxy when considering non-stationary observation noise. (a) shows the estimated distance field with the observation parameters tuned for the least noisy of the wall observations. (b) shows the error compared with the ground truth. (c) is the uncertainty proxy introduced in [5]. (d) is the proposed uncertainty proxy which patterns match the area of low and high error in (b).

closest map voxel. Then, as the GP is computed locally, the integral over the sphere around $\mathbf{s}(\mathbf{x})$ can be replaced by the integral over the full $\mathbb{R}^3$ space. As the integral over $\mathbb{R}^3$ of the kernel function is a constant c, $\nu(\mathbf{s}(\mathbf{x})) \approx c \sum_i \alpha_i$, with $\boldsymbol{\alpha} = (\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma\mathbf{I})^{-1}\mathbf{1}$ (already computed in (8)). In the end, the additional computation cost of our novel uncertainty proxy is negligible. An example of the proposed proxy is given in Fig. 6 (d). Note that in its current state, the proxy does not directly provide the standard deviation of the inferred distance. We will investigate this in our future work to give ϕ a physical meaning.

D. Scan-to-map registration

Before inserting a point cloud in the map, the proposed mapping algorithm performs "scan"-to-map registration using the aforementioned distance function $d(\mathbf{x})$ in a similar way to [7]. The point cloud registration consists of iteratively minimising the point-to-map distance queries for each of the cloud's points. Conceptually, this approach performs association-free registration thanks to the continuous nature of the distance field. Formally, the sensor pose is estimated by solving the following non-linear least-square problem:

$$\mathbf{T}_W^{L_{\tau_i}*} = \underset{\mathbf{T}_W^{L_{\tau_i}}}{\operatorname{argmin}} \sum_{j \in \mathcal{P}_i} \left(d\left(\mathbf{T}_W^{L_{\tau_i}} \begin{bmatrix} \bar{\mathbf{x}}_j \\ 1 \end{bmatrix} \right) \right)^2. \quad (9)$$

This problem is solved with the Levenberg-Maquardt algorithm accounting for the SE(3) nature of $\mathbf{T}_W^{L_{\tau_i}}$.

E. Free-space carving

As explained earlier in this paper, the point filtering step is designed to remove unreliable lidar points including

Fig. 7. Illustration of the free-space carving process of the proposed mapping framework (using [13]'s living room simulated environment).

the ones belonging to dynamic objects. Occasional failure of the filtering can lead to unwanted points in the map. Moreover, if we consider objects in the environment that are static at a given time but removed from the scene at a later time, we want a mechanism to update/clean the map accordingly. Under the name of *free-space carving* the proposed mapping algorithm remove unwanted cells from the map before inserting a novel scan to it.

The principle of the proposed mechanism relies of a comparison between the spherical projections of the current scan and the map cells in the vicinity of the current sensor position. First, the scan is projected onto an image-like data structure by keeping the point with the smallest range in each of the pixels. Then, the map points present in a radius around the sensor position are queried and projected in the aforementioned image-like data structure. If a map point is projected in a pixel that has a larger range (minus a threshold), it should be removed from the map as it occurred between a surface currently observed and the sensor. Fig. 7 illustrates this process.

F. Meshing

Once the lidar data collection finishes, we aim at providing a clean map of the environment by meshing our sparse voxelised map with screened Poisson surface reconstruction [9]. This method requires a point cloud and the associated

oriented normals as input. Given the proposed voxelised map of the environment, the point cloud of the map is readily available as the centroid of each of the cells. For the normals, we leverage the GP-based distance field. Atop the distance field $d(\mathbf{x})$, our GP-based map allows the direct inference of the gradient of $d(\mathbf{x})$ thanks to the use of linear operators on the GP kernel [14]. By inferring $\nabla d(\mathbf{x})$ for each cell's centroid, we obtain the corresponding normal vectors. However, as the proposed distance field is not signed, the normals are not necessarily consistent throughout the environment. Thus, the direction of the normal vectors is corrected using the position of the sensor when it observes the points in each cell.

As per its original design, the screened Poisson reconstruction aims at computing a closed mesh. Such a feature is not necessarily convenient for robotic applications where systems mostly deal with partial observations of the environment. This leads to "hallucinations" that erroneously fill with triangles in areas where no data have been collected. To avoid this, we perform a simple mesh-cleaning operation that consists of removing from the mesh every triangle whose centroid or corners are too far from the surface according to our distance field. Note that the Poisson surface reconstruction builds an indicator function similar to a truncated signed distance field that crosses a level set on the surface, before running a marching cube algorithm. In future work, we will investigate how to elegantly obtain the level set crossing based on our GP-based representation of the environment.

V. IMPLEMENTATION

In this section, we present and discuss some specific details of our ROS-based [15] C++ implementation.

A. Low-level data structures

1) Hashmaps: All the hashmaps used in the work (for voxel management in the point filtering and the global map) are using the `ankerl::unordered_dense::map` implementation [16]. In [17], the author benchmarks numerous C++ hashmap implementations and shows that theirs significantly outperforms the one from the standard library. The values of the hashmap are stored through pointers to ensure the compactness of the hashmap storage.

2) Spatial indexing: The constraints on the choice of the spatial index are as follows:

- Fast and scalable insertion of new elements.
- Possibility to remove points without the need to recreate the index.
- Allowing for N-closest neighbour and radius search.

Both *ikd-Trees* [18] and *PH-Trees* [19] match the aforementioned requirements. We have conducted simple toy-example evaluations of both methods and observed a significant advantage for the PH-Tree implementation available at [20].

For efficiency, the spatial indexing structure is not updated with the changing centroid of the voxels but keeps using the coordinates of the first point that occurred in each voxel. Accordingly, the closest neighbour or radius searches performed with the "out-of-date" spatial index are only

(a) Sequence `quad_easy`   (b) Sequence `quad_hard`   (c) Sequence `cloister`

Fig. 8. Top-down visualisation of the final mesh map using 2FAST-2LAMAA on several sequences of the *Newer College Dataset* [3]
.

approximations. To address this, a distance query in our map structure involves a K closest neighbours and computes the GP-based distance field with the K voxels independently. The distance returned is the smallest of the K resulting distance values.

B. Non-linear optimisations

All the optimisations in this work (lidar-inertial undistortion and scan-to-map registration) are based on the open-source *Ceres* non-linear least-square solver [21]. In the case of the scan-to-map registration (9), we use a Cauchy loss function to attenuate the impact of outliers. The analytical Jacobians of the residuals are provided to the solver. To lower the computational cost of the overall pipeline we adopted a basic key-framing strategy to perform (9) only when the sensor has moved sufficiently or after a fixed time period.

C. ROS nodes

The open-source ROS1 and ROS2 code of our implementation is organised in several nodes:

- `scan_maker`: A simple node to reorganise the raw incoming lidar data into "scans" as defined in this paper.
- `lidar_feature_detection`: This piece of code performs the feature point detection on each of the previously constituted scans.
- `lidar_scan_odometry`: Here the IMU-aided scan undistortion and the unreliable point filtering are performed.
- `gp_map`: This node contains our efficient GP-based distance field representation. The scan-to-map registration, free-space carving mechanism, and triangle mesh reconstruction are all part of this node.
- `field_visualiser`: A simple visualisation node to demonstrate queries of our distance-field map via calls to the `QueryDistField` service. The returns are published as a distance-coloured `PointCloud2`.

VI. EXPERIMENTS

We have tested the proposed framework on the `quad` and `cloister` sequences of the *Newer College Dataset* [3].

Node	CPU load [%]	RAM usage [MB]
Scan maker	1.81	61
Feature detection	1.27	41
Scan odometry	5.36	918
GP map	14.11	737

TABLE I

COMPUTATIONAL LOAD OF THE DIFFERENT NODES OF
2FAST-2LAMAA.

The data was collected with a 128-beam Ouster lidar and its embedded 6-DoF IMU. Note that the dataset also provides visual information from multiple cameras. That information is not used in the proposed pipeline but we display it in our qualitative results to give additional context information to the reader. Fig. 8 shows the obtained map overview with the `quad_easy`, `quad_hard`, and `cloister` sequences, while Fig. 9 provide closer viewpoints. As illustrated further in Fig. 10, one can see that despite people moving around the environment, the resulting map is free from any 3D trails typically left by moving objects in lidar-built maps.

All the experiments are run in real-time on a consumer-grade laptop equipped with an i7-1370p CPU and 32GB of RAM. In Table I, we provide the CPU and RAM usage for the different nodes of the pipeline on the `cloister` sequence. Note that the triangle mesh reconstruction is not considered there as it is a one-off computation automatically performed at the end of the data recording. The time consumption for the reconstruction is detailed in Table II. Regarding downstream application uses of the distance field, the query time for a single point is around $250\mu s$. However, when querying multiple points and using some caching (GP computation reuse), the average per-query time is around $5\mu s$.

In order to compare with other distance field frameworks such as [22] which assumes already registered scans, we provide the average computation time per scan (keyframes) with and without the registration. The results are shown in Table III. We have also included a version of the registration

Camera image (not used)   **Final map view**

quad_easy (frame 0)

quad_easy (frame 321)

quad_hard (frame 127)

quad_hard (frame 342)

cloister (frame 552)

cloister (frame 1899)

Fig. 9. Visualisation of the 2FAST-2LAMAA mesh maps associated with camera image for reference (not used in the estimation pipeline).

Reconstruction time [s]	
Normal queries	6.52
Screened Poisson	19.77
Mesh cleaning	3.48
Writing to disk	0.68
Total	30.45

TABLE II

TIME CONSUMPTION OF THE TRIANGLE MESH SURFACE RECONSTRUCTION.

Version	Time[ms]
GP reg. + free-space carving + map update	277
Approx. reg. + free-space carving + map update	143
Free-space carving + map update	73.2
Map update	15.2

TABLE III

PER-SCAN TIME CONSUMPTION OF GP MAPPING NODE.

that only uses the point-to-point distances directly out of the spatial indexing without the use of the GP-based distance field in the cost function (9) (the data structure still allows for distance field queries though). When compared with the benchmarked methods in [22], the proposed implementation (free-space carving and map update) is faster by almost an order of magnitude. Note that the free-space carving and map update are both performed in a single thread. It makes the proposed mapping node attractive for low-power embedded systems.

VII. CONCLUSION

In this document, we have presented a novel lidar-inertial framework named 2Fast-2Lamaa for localisation and mapping in dynamic environments. It builds on lidar-inertial undistortion of the sensor data through the estimation of the system's velocity, gravity direction, and inertial biases. This is followed by a global registration step using a continuously incremented GP-based distance field. The soundness of 2Fast-2Lamaa is shown over sequences from the *Newer College Dataset* [3] and we have provided detailed information about the computation requirements.

In the future, we will perform a thorough evaluation of 2Fast-2Lamaa over a wide range of dataset. Additionally, as the method performs scan-to-map registration with a single global map without storing the past trajectory, it does not possess a mechanism to integrate explicit loop-closure constraints. While not required for relatively small environments as shown in this document, it is crucial to address the unavoidable drift over large-scale datasets. Thus, the use of a sub-mapping strategy and graph-based pose estimation will be part of our future work.

REFERENCES

[1] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "3D Lidar-IMU Calibration based on Upsampled Preintegrated Measurements for Motion Distortion Correction," *IEEE International Conference on Robotics and Automation*, 2018.

[2] ——, "IN2LAAMA: INertial Lidar Localisation Autocalibration And MApping," *IEEE Transactions on Robotics*, 2021.

[3] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-camera lidar inertial extension to the newer college dataset," 2021.

[4] R. Falque, C. Le Gentil, and F. Sukkar, "Dynamic object detection in range data using spatiotemporal normals," in *Australasian Conference on Robotics and Automation*, 2023.

[5] C. Le Gentil, O.-L. Ouabi, L. Wu, C. Pradalier, and T. Vidal-Calleja, "Accurate gaussian-process-based distance fields with applications to echolocation and mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1365–1372, 2024.

[6] C. Le Gentil, R. Falque, and T. Vidal-Calleja, "Real-time truly-coupled lidar-inertial motion correction and spatiotemporal dynamic object detection," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

| (a) Camera image (not used) | (b) Final map viewpoint A | (c) Final map viewpoint B |

Fig. 10. Visualisation of the effectiveness of dynamic object removal (leveraging both the proposed point cloud filtering and free-space carving mechanism) at visual frame 778 of quad_easy. Humans walking around the environment are not present in the final map.

[7] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, "Log-gpis-mop: A unified representation for mapping, odometry, and planning," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 4078–4094, 2023.

[8] K. M. B. Lee, Z. Dai, C. L. Gentil, L. Wu, N. Atanasov, and T. Vidal-Calleja, "Safe bubble cover for motion planning on distance fields," 2024. [Online]. Available: https://arxiv.org/abs/2408.13377

[9] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, jul 2013. [Online]. Available: https://doi.org/10.1145/2487228.2487237

[10] C. Le Gentil and T. Vidal-Calleja, "Continuous latent state preintegration for inertial-aided systems," *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 874–900, 2023.

[11] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, sep 1975.

[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[13] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[14] S. Särkkä, "Linear operators and stochastic partial differential equations in Gaussian process regression," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 151–158, 2011.

[15] O. S. R. Foundation, "ROS - Robot Operating System." [Online]. Available: https://www.ros.org/

[16] M. Leitner-Ankerl, "A fast & densely stored hashmap and hashset based on robin-hood backward shift deletion." [Online]. Available: https://github.com/martinus/unordered_dense

[17] ——, "Comprehensive C++ Hashmap Benchmarks 2022." [Online]. Available: https://martin.ankerl.com/2022/08/27/hashmap-bench-01/

[18] Y. Cai, W. Xu, and F. Zhang, "ikd-tree: An incremental kd tree for robotic applications," *arXiv preprint arXiv:2102.10808*, 2021.

[19] T. Zäschke, C. Zimmerli, and M. C. Norrie, "The ph-tree: a space-efficient storage structure and multi-dimensional index," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 397–408. [Online]. Available: https://doi.org/10.1145/2588555.2588564

[20] T. Zäschke, "PH-Tree C++ implementation." [Online]. Available: https://github.com/tzaeschke/phtree-cpp

[21] S. Agarwal and K. Mierle, "Ceres Solver." [Online]. Available: http://ceres-solver.org

[22] L. Wu, C. L. Gentil, and T. Vidal-Calleja, "Vdb-gpdf: Online gaussian process distance field with vdb structure," 2024. [Online]. Available: https://arxiv.org/abs/2407.09649