2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Lifelong Localization in Dynamic Indoor Environments Combining Odometry with Sparse Distance Sampling

Michael M. Bilevich† Tomer Buber† Dan Halperin†

***Abstract*— Localization is a key task in robot navigation, and many techniques exist for it. In many plausible scenarios, a robot might face unforeseen, dynamic obstacles, rendering any pre-determined map inaccurate for localization. In this work, we propose a robust lifelong localization framework in dynamic planar indoor environments, using the robot's odometry and sparse distance sampling. We demonstrate how distance samples can be used to provide a robust prior on the robot's location. This technique can solve the kidnapped robot problem in real time, up to symmetries. Based on insights from real-world recorded data, we also account for dynamic obstacles. We then fuse this prior, over time, with the odometry to converge to the robot's location. A central property of our method is that it provably converges to the robot's ground truth pose even in large indoor environments when the environment is static. We further show that this guarantee also holds in dynamic environments, as long as the nature of those changes has been correctly learned. We demonstrate the effectiveness of our approach in different real-world indoor environments. In particular, we achieve a localization comparable to SLAM with merely a few (sixteen) distance samples, as opposed to the full LiDAR range. Sufficing with only sparse distance sampling is advantageous in terms of sensor cost, privacy, storage space, and transmission bandwidth.**

I. Introduction

Robot localization is a critical ingredient in robot navigation [1]; even if the map of the environment is entirely known, and we plan a collision-free path of motion, we still need to determine the robot's location in the environment to carry out that path of motion. Determining the location of a robot in the environment is referred to as localization. Localization has been extensively studied and can be solved with various techniques and sensors. These methods may be intrinsic (attached to the robot) or extrinsic (attached to the environment).

Extrinsic approaches are commonplace; one well-known example is the Global Positioning System (GPS) [2]–[4]. By priorly placing landmarks in designated locations in the environment, the robot can triangulate its location with respect to the landmarks it perceives. These landmarks can be sophisticated and utilize properties of electromagnetic waves (such as RSSI [5, 6] and RFID [1]), or they can be as simple as printed QR codes [7]–[9]. Methods originally developed for virtual reality applications can also be used for localization [10]–[12]. A disadvantage of the extrinsic approach is that prior placement of landmarks may not be viable in some environments, such as confined environments (mines, caves) [13, 14] or disaster-stricken environments (during fires or earthquakes) [15, 16].

†Blavatnik School of Computer Science and Artificial Intelligence, Tel-Aviv University, Israel. This work has been supported in part by the Israel Science Foundation (grant no 3598/25), by the Blavatnik Computer Science Research Fund, and by the Shlomo Shmelzer Institute for Smart Transportation at Tel Aviv University.

Fig. 1. A demonstration of our algorithm in different real-world scenarios. All images are screenshots from RViz. In all scenarios, the robot uses merely $k = 16$ distance measurements and its odometry to consistently find its location. All colorful squares are pose candidates returned by our method, where their color ranges from red (least likely) to magenta (most likely). Cyan marks the ground truth location. **Top**: A large-scale floor plan (`fl4`). **Bottom left**: A map of our laboratory (`lab446`). **Bottom right**: A floor plan of an apartment (`apt`).

On the other hand, intrinsic localization is performed by sensors mounted on the robot itself. In such a case, we assume that we are given a map of the environment or that the sensors are sufficiently sophisticated to both map the environment and track the robot's location in the (simultaneously) mapped environment. The latter is the task of the intensively investigated Simultaneous Localization and Mapping (SLAM) [17]–[20]. Usually, the robot is mounted with depth cameras or LiDAR sensors. Both may be expensive [21], especially when considering swarms of robots. Furthermore, if the computations are performed on a remote computer, transmitting LiDAR scans or image data over the network requires high bandwidth, which may cause communication interference [22]. One primary concern regarding using cameras attached to the robots is the potential breach of privacy [23, 24].

The broad term *localization* can refer to different problems, sometimes leading to vastly different solutions. One example is the *kidnapped robot problem* [25], where we need to determine the location of a robot given the map of the environment, but without any prior knowledge of its location. On the contrary, there is the *tracking problem* [5], where we start with a known robot location and strive to fix any drift and measurement errors throughout its motion. This is also known

as *dead reckoning* [26]–[29]. Life-long localization [30]–[34] is the problem of determining the localization of a robot in a known environment throughout multiple sessions. Usually, this is performed by storing and updating the map [33, 34] during multiple sessions. Life-long indoor localization poses many difficult challenges, such as overcoming the presence of dynamic obstacles and changes in the environment [35, 36], as well as computational [33] challenges of effectively and quickly maintaining and refining the maps.

In this work, we present a lifelong indoor localization technique that requires merely odometry and a sparse distance sampling. We will also refer to this method as *sparse distance sampling localization (SDSL)*, since we use only a few distance samples, as opposed to a range of distance samples returned by, for example, a LiDAR sensor.

Recent works [37]–[42] have dealt with intrinsic localization in already-known environments. These methods are deterministic, unlike more classical approaches such as Bayesian filters and Monte-Carlo localization [6, 43]. These works are robust for different kinds of errors, such as errors in measurements and mapping. We note that [37]–[39] used the full range of distance measurements given by a LiDAR sensor, while our works [40]–[42] used only a sparse distance sampling.

We improve upon [40, 42] by incorporating a probabilistic machinery, inspired by Bayesian localization, to utilize the robot's odometry. We also account for unforeseen dynamic obstacles and changes in the environment, based on the insight presented in [41].

We assume that the environment is a subset of $\mathbb{R}^2$, and our robot is *planar* with three degrees of freedom - two for translations in the environment, and one for rotation about its origin. The robot is equipped with an array of range sensors with known offsets from the origin and any odometry sensor (e.g., an inertial measurement unit or IMU, wheel encoder, or optical flow). We assume that we are given a map that is a good approximation of the environment, albeit it may have some topological errors and unforeseen dynamic obstacles in the environment.

Similar to previous works, this framework is beneficial in terms of cost, privacy, and transmission bandwidth.

See Figure 1 for an illustration of our method.

A. Contribution

Our contributions are as follows:

- An effective lifelong localization technique for planar robots, fusing odometry dead reckoning with sparse distance sampling, which is inherently robust for the kidnapped robot problem even in dynamic environments.
- Guarantees on the correctness and completeness of the algorithm, proving that our localization technique would always report a pose candidate close to the true location, when the dynamic nature of the environment is reasonably predictable.
- We demonstrate our method on a mobile planar robot, showing the potential for real-world application.
- An open-source C++ library with Python bindings [44] that can utilize multi-core CPUs, as well as ROS2 [45] packages for easy deployment on general robots.

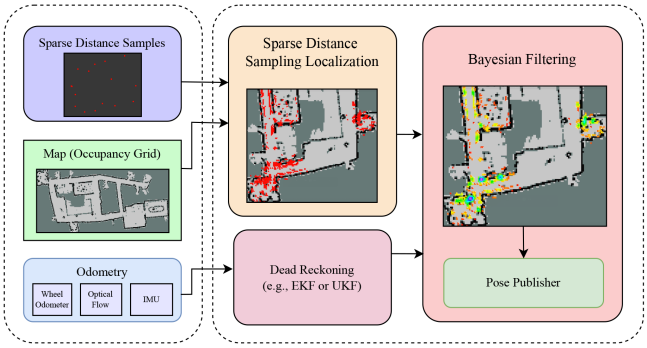

Fig. 2. An overview of our method. On the left-hand side are the inputs to our method; we assume that the map is pre-determined, and we receive a sparse (e.g., $k=16$) distance sampling, and any kind of odometry. We use odometry to calculate the dead reckoning of the robot. We also match the distance samples (SDSL) with the map to get the set of all possible pose candidates. We then fuse the dead reckoning with the SDSL to get the belief state for our robot, which is the likelihood of each pose candidate. Once there is one pose that is more likely than the others, we publish it as the robot's location.

II. Preliminaries and Problem Statement

A. Subdivision Search of Fibers

A fiber of some function f, denoted by $f^{-1}(y)$, is the set of all values x such that $f(x)=y$, given a value y. A key observation of [40, 42] is that given the distance measurements, the set of all possible locations is the intersection of fibers of the distance function. This task is studied in computer graphics, computational geometry, and computer algebra [46]. One common example for the search for fibers is the meshing of implicit surfaces, which can be carried out by a variety of popular techniques: grid-based methods like marching cubes and dual contouring [47]–[50] or Delaunay refinement [51]–[53]. Another common technique of searching and representing implicitly defined geometry is by subdivision searches, i.e., via quadtrees for two dimensions, octrees for three dimensions, and orthrees for higher dimensions [54]–[56]. The method presented in this work is also inspired by methods of the soft subdivision search (SSS) framework [57]–[59]. Generally, subdivision methods are superior to grid-based methods in terms of running time complexity [60].

B. Problem Statement

In this work, we assume that the environment $\mathcal{W}\subset\mathbb{R}^2$ is a closed subset of the plane. The robot can translate and rotate freely in the environment, and its configuration space [61] is $\mathcal{C}:=\mathbb{R}^2\times\mathbb{S}^1$. For a configuration $q=(q_p,q_\theta)\in\mathcal{C}=\mathbb{R}^2\times\mathbb{S}^1$, q_p is the position of the robot's origin in the plane and q_θ is its orientation.

Note that the orientation space can be represented by the angles $\mathbb{S}^1\simeq[0,2\pi)$ or the corresponding points on the unit circle $\mathbb{S}^1\subset\mathbb{R}^2$. In this work, we use both notions interchangeably.

The robot is mounted with k range sensors, with known offsets $g_1,\ldots,g_k\in\mathbb{R}^2\times\mathbb{S}^1$; Initially, the sensors are mounted at an offset g_{i_p} from the robot's origin and point in the direction g_{i_θ}.

Assuming that the robot is at some configuration $q\in\mathcal{C}$, when measuring distance from the ith sensor, we cast a ray emanating from the sensor's current position, which is the offset g_{i_p} translated and rotated by q. The ray's direction

is the direction $g_{i\theta}$ rotated by q. We denote the application of the pose q on the sensor position g_{i_p} and direction $g_{i\theta}$ by $q \cdot g_{i_p}$ and $q \cdot g_{i\theta}$, respectively.

Let $h : \mathbb{R}^2 \times \mathbb{S}^1 \to \mathbb{R}_{\geq 0}$ denote the distance measurement function. Then $h(p, \theta)$ is the distance between the point $p \in \mathbb{R}^2$ and the first intersection of a ray emanating from p in the direction $\theta \in \mathbb{S}^1$ with the boundary of the environment $\partial\mathcal{W}$. The distance measurement of the ith sensor, denoted by $d_i \geq 0$, is $d_i = h(q \cdot g_{i_p}, q \cdot g_{i\theta})$.

Problem Statement: Given a map of the environment $\mathcal{W}$, the sensor offsets g_i, and their corresponding distance measurements d_i for $i = 1,...,k$, find the set of all poses $q \in \mathcal{C}$ that, by placing the robot at pose q, it would measure the distances $d_1,...,d_k$ at the offsets $g_1,...,g_k$, respectively.

III. Method Overview

We now provide an overview of our method. See Figure I for an illustration of the proposed approach. In the figure, we use ROS2 [45] notions throughout, and ROS was our choice for the implementation. However, the method and algorithms can be implemented with any other framework. Furthermore, as described in Section VII-A, our core method is implemented as a standalone C++ library independent of ROS.

A. Sensors Input and Pre-determined Map

Our method builds on three main inputs. The first is a pre-existing map of the environment, which provides the global frame of reference. The second is the robot's odometry, giving a continuous estimate of its motion over time. The third is a sparse set of k range measurements, for which the offsets $g_1,...,g_k$ are known and pre-determined. No additional sensing or framework-specific assumptions are required.

B. Message Processing and Sensor Fusion

Our approach works by first processing each modality (the odometry, dead reckoning, and the SDSL) separately and efficiently. Then, we combine (fuse) both intermediate results into a more robust localization. Using the robot's odometry, we can compute dead reckoning. That is, by integrating the odometry, we can estimate the trajectory/pose of the robot with respect to its origin of motion [28]. This estimate suffices locally, but may drift for longer horizons due to factors like wheel slip and sensor noise. Note that this origin is not necessarily the origin of the map. Importantly, this local motion frame is not necessarily aligned with the global map frame, so an additional transformation between the two must eventually be established.

With sparse distance sampling, our method produces a set of candidate poses that capture all robot locations consistent with the measurements in the global map.

Since the number of samples is small and the environment may include moving obstacles, the resulting set of poses can spread over vastly different areas of the map. These candidate poses are similar to the particles in Monte Carlo localization. However, they are computed deterministically, based on the geometric constraints posed by the distance measurements. Details of the sparse distance sampling localization technique are presented in Section IV

We then fuse the odometry trajectory with the poses set returned by the SDSL method. When using both, the odometry provides smooth local motion, and the sparse samples keep the estimate tied to the global map. The result is a weighted set of poses that reflects the robot's most likely location. Details of the fusion process are given in Section V.

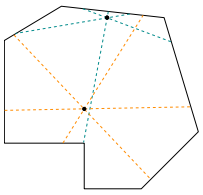

Fig. 3. Illustration of two poses with $k = 6$ distance measurements. The top point (whose rays are cyan) is closer to the environment's wall than the bottom (whose rays are orange). Notice how the bottom point is more likely to see dynamic obstacles, as there is more space between the robot and the environment. For the top point, about half of its rays measure the wall, which is relatively close.

C. Method Output

Finally, we must choose the robot's most probable (single) location. To do so, we cluster the particles and weight the probability/likelihood of each cluster. Whenever a cluster is more likely than the rest, its center of mass (weighed by the poses' likelihood) is chosen as a pose estimate, and the transform from the `odom` frame to the `map` is updated accordingly.

IV. SDSL for Kidnapped Robot Problem

In this section, we describe the Sparse Distance Sampling Localization (SDSL) technique, which outputs the set of all feasible robot poses, given a single sparse distance sampling, i.e., a set of k measurements $d_1,...,d_k \geq 0$ and their corresponding pre-determined offsets $g_1,...,g_k \in \mathbb{R}^2 \times \mathbb{S}^1$.

To do so, we perform a subdivision search, similar to the one described in detail in [42, 60]. The derivations are in particular identical to [42], and thus are omitted here for brevity. We also note that, by definition (see Section II-B), for a single distance measurement d_i, the set of all possible locations that satisfy that measurement is a *fiber* of the distance measurement function.

We choose a parameter $\delta > 0$, which is our desired approximation precision, and we also assume we are given $\varepsilon > 0$ as a distance measurement error bound. We start with a bounding volume V_0 of the configuration space—as one voxel, and recursively subdivide a voxel into smaller voxels only if it may contain the ground truth result, until all voxels are of diameter $< \delta$. Thus, the crux of our method is effectively and correctly determining whether a voxel may or may not contain a ground truth result.

Furthermore, improving upon [42], we assume that there is a *k-k'-dynamic gap* [41]. That is, we assume that out of the k distance measurements, at least k' samples correspond to features in the pre-determined map. Of course, this k-k' gap depends on the dynamic nature of the environment, the semantic context of the present time (e.g., holidays or nights may have less traffic of dynamic obstacles), and even the location in the room; one possible example is that poses close to walls are almost surely guaranteed to have $k' > k/2$, regardless of the dynamic nature. See Figure 3 for an illustration.

Algorithm 1 describes our overall method. For each voxel, we estimate the value k'. The method VoxelXPred

(for Voxel intersection Prediction), tests whether a single measurement d_i and its corresponding offset g_i "agree" with the environment. That is, whether there exists some pose in the voxel for which, by placing the robot at that pose, the sensor whose initial offset is g_i would measure the distance d_i (up to an error of ε). If at least k' measures do not agree with the environment, the voxel is discarded; otherwise, the voxel is split (or reported, if its diameter is less than δ).

To evaluate VoxelXPred, we merely intersect an axis-aligned box with the boundary of the environment $\mathcal{W}$. Denote by $F_{d,g}:\mathcal{C}\to\mathbb{R}^2$ the function that takes a robot configuration q, and returns the point in $\mathbb{R}^2$ which is d-units forward from the sensors g when the robot is at pose q. Note that if the intersection $F_{d,g}(V)\cap\partial\mathcal{W}\neq\emptyset$ is not empty, there exists some pose in that voxel V that, by placing the robot in that pose, the sensor at offset g would measure exactly d. Although the geometry of the set $F_{d,g}(V)$ can be precisely described and computed, we approximate it by taking its bounding box, which is simpler to compute. More details can be found in [42], and the pseudo-code is presented in Algorithm 2.

The method $\mathrm{Split}(V)$ splits a voxel into a collection of smaller voxels such that their union equals the original voxel V. In this work, we split the voxel into 2^3 sub-voxels by splitting into two in each dimension. The method $\mathrm{EstimateKPrime}(V)$ is described in Section IV-A.

Finally, we coincide the voxel outputs with a point cloud of the voxel centers. This can be done by choosing a value of δ which is strictly smaller than the desired accuracy, such that each two poses in the same voxel are effectively indistinguishable.

Algorithm 1 Sparse Distance Sampling Localization

Require: $\mathcal{W}\subset\mathbb{R}^2, d_1,...,d_k\geq 0, g_1,...,g_k\in\mathbb{R}^2\times\mathbb{S}^1$
Require: $\delta>0,\varepsilon>0$
Require: $V_0\subset\mathcal{C}, \mathcal{W}\times\mathbb{S}^1\subseteq V_0$
  $\mathcal{Q}\leftarrow\{V_0\}$, $\mathcal{Q}'\leftarrow\emptyset$
  while $\max_{V\in\mathcal{Q}}\mathrm{diam}V\geq\delta$ **do**
    for $V\in\mathcal{Q}$ **do**
      $k'\leftarrow\mathrm{EstimateKPrime}(V)$
      $cnt\leftarrow 0$
      for $i=1,...,k$ **do**
        if $\mathrm{VoxelXPred}(\mathcal{W},\mathrm{d_i},\mathrm{g_i},\mathrm{V},\varepsilon)$ **then**
          $cnt\leftarrow cnt+1$
        end if
      end for
      if $cnt\geq k'$ **then** $\mathcal{Q}'\leftarrow\mathcal{Q}'\cup\mathrm{Split}(V)$
      end if
    end for
    $\mathcal{Q}\leftarrow\mathcal{Q}'$, $\mathcal{Q}'\leftarrow\emptyset$
  end while
  return $\mathcal{Q}$

A. Estimating the Expected Value of k'

Given a voxel V, we need to provide a reasonable estimate of the value of k', so we will not miss the ground truth location. However, we need a value that is still close enough to k so that the resulting set $\mathcal{Q}$ will be sufficiently small.

As stated in Section IV, this value of k' may depend on various factors. In this work, we suggest a heuristic,

Algorithm 2 $\mathrm{VoxelXPred}(\mathcal{W},d,g,V,\varepsilon)$

Require: $V=[q_{\mathrm{bl}},q_{\mathrm{tr}})\subset\mathcal{C}$, $\mathcal{W}\subset\mathbb{R}^2, g\in\mathbb{R}^2\times\mathbb{S}^1, d\geq 0,\varepsilon>0$
  $\alpha_1\leftarrow g_x+d\cdot\cos g_\theta$, $\alpha_2\leftarrow g_y+d\cdot\sin g_\theta$
  $\beta_1\leftarrow\tan^{-1}(-\alpha_2/\alpha_1)$, $\beta_2\leftarrow\tan^{-1}(\alpha_1/\alpha_2)$
  $\theta_1\leftarrow q_{\mathrm{bl}\theta}$, $\theta_2\leftarrow q_{\mathrm{tr}\theta}$, $\Theta\leftarrow\{\theta_1,\theta_2\}$
  for $j=-3,...,3$, $i=1,2$ **do**
    ▷ In fact, only few j values are feasible
    if $\theta_1\leq\beta_i+j\cdot\pi\leq\theta_2$ **then**
      $\Theta\leftarrow\Theta\cup\{\beta_i+j\cdot\pi\}$
    end if
  end for
  $x_1\leftarrow\infty, x_2\leftarrow-\infty$, $y_1\leftarrow\infty, y_2\leftarrow-\infty$
  for $\theta\in\Theta$ **do**
    $\gamma_1\leftarrow\cos\theta\cdot\alpha_1-\sin\theta\cdot\alpha_2$, $\gamma_2\leftarrow\sin\theta\cdot\alpha_1+\cos\theta\cdot\alpha_2$
    $x_1\leftarrow\min\{x_1,\gamma_1\}$, $x_2\leftarrow\max\{x_2,\gamma_1\}$
    $y_1\leftarrow\min\{y_1,\gamma_2\}$, $y_2\leftarrow\max\{y_2,\gamma_2\}$
  end for
  $x_1\leftarrow x_1+q_{\mathrm{bl}x}-\varepsilon$, $x_2\leftarrow x_2+q_{\mathrm{tr}x}+\varepsilon$
  $y_1\leftarrow y_1+q_{\mathrm{bl}y}-\varepsilon$, $y_2\leftarrow y_2+q_{\mathrm{tr}y}+\varepsilon$
  $B\leftarrow\big[(x_1,y_1),(x_2,y_2)\big)\subset\mathbb{R}^2$ **return** $B\cap\mathcal{W}\neq\emptyset$

which looks only at the distance $d\left((q_x,q_y),\partial\mathcal{W}\right)\geq 0$ of a configuration $q=(q_x,q_y,q_\theta)\in\mathcal{C}$ from the boundary of the environment $\partial\mathcal{W}$. We note that the distance from the boundary is an invariant feature under translations and rotations of the map, and should be at least comparable for environments whose rooms are of similar dimensions.

Based on data collection from real-world environments, which is described in Section VII-D, we implement the following heuristic function $\tilde{f}_{k'}:[0,\infty)\to[0,1]$ which returns the expected *ratio* k'/k of k' over k for a pose whose distance from the boundary is $\mathrm{dist}=d((q_x,q_y),\partial\mathcal{W})\geq 0$:

$$\tilde{f}_{k'}(\mathrm{dist})=\begin{cases}0.8 & \text{if } \mathrm{dist}<0.975[m]\\ 0.7 & \text{else}\end{cases}. \tag{1}$$

As shown in Section VII-D, this simple function is a good rough approximation for typical office environments, and we see that it also performs well in practice.

To evaluate $\tilde{f}_{k'}$ on a voxel, we take the minimal value of $\tilde{f}_{k'}$ of each of its vertices and its center point.

V. Fusing SDSL with Odometry

In this section, we describe how the lifelong dead reckoning pose estimate can be combined with the point cloud returned by the SDSL algorithm (Algorithm 1), using Bayesian filtering [62].

Denote by $\hat{\gamma}:[0,\infty)\to\mathcal{C}$ the estimated path, returned by any dead reckoning localization technique. This $\hat{\gamma}$ is estimated from the robot's odometry. We assume that for each time t, we can query the value of $\hat{\gamma}(t)$.

We expect a lifelong stream of consecutive runs of the SDSL algorithm, i.e., a sequence $X_1,X_2,...$ where each $X_n=\{q_1^{(n)},...,q_{N_n}^{(n)}\}\subset\mathcal{C}$ is a set of N_n configurations, returned by Algorithm 1 at time t_n. For clarity, we introduce the following notation: $q_i^{(n)}$ denotes the i-th candidate pose in X_n. Denote the following estimated pose difference:

$$U_n=\hat{\gamma}(t_n)\cdot\left(\hat{\gamma}(t_{n-1})\right)^{-1}, \tag{2}$$

which is the estimated transformation the robot performed from time t_{n-1} to t_n, and can be thought of as simply the robot's odometry. Similar to Monte Carlo localization [63], we can use these odometries $U_1, U_2, ...$ to evaluate the likelihood, over time, of each pose. The main difference is that instead of stochastically resampling, at each iteration n, we recompute the set of all possible poses, using the geometric constraints imposed by the perceived distance measurements, via the SDSL method. In Section VI, we also show that this approach has rigorous theoretical merits.

In Algorithm 3 we present the pseudo-code for the procedure that computes the *belief* state, denoted by $\mathrm{Bel}(\cdot)$, which computes for each pose $q_i^{(n)}$ its likelihood $\mathrm{Bel}(q_i^{(n)})$.

From the recursive formulation of Bayesian filtering, we recall that the belief state can be recursively enumerated [62]:

$$\mathrm{Bel}(X_n) \propto \int_{X_{n-1}} \mathbb{P}[X_n | X_{n-1}=q, U_n] \cdot \mathrm{Bel}(q) dq. \quad (3)$$

Note that since we took into account the distance measurements in the SDSL step, we ignore them here. Equation 3 omits the denominator that usually appears in the Bayes filter recursion, as we can simply normalize the probabilities at each time step n.

At time $n=1$, the belief state is set as a uniform distribution, and all poses have the same likelihood. We model the transition $\mathbb{P}[X_n|U_n, X_{n-1}=q]$ as a normal distribution, with a standard deviation of ε. That is, $X_n \sim \mathcal{N}(U_n \cdot X_{n-1}, \varepsilon^2)$.

The method $\mathrm{Normalize}(\mathrm{a}_1,...,\mathrm{a}_\mathrm{m})$ divides each element by the sum of all elements, i.e., $a_i \leftarrow a_i / \sum_{j=1}^{m} a_j$.

We also note that instead of directly computing the exponential Gaussian probability, we can instead use the *log likelihoods* [43]. For simplicity, Algorithm 3 shows the straightforward approach.

Algorithm 3 Fusing SDSL with Odometry

Require: $X_1, X_2, \cdots \subset \mathcal{C}$, $U_1, U_2, \cdots \in \mathcal{C}$
  $p \leftarrow \{\}$
  for $i=1,...,N_1$ **do** $p(q_i^{(1)}) = \frac{1}{N_1}$
  end for
  for $n=2,...$ **do**
    for $q_i^{(n)} = q_1^{(n)}, ..., q_{N_n}^{(n)} \in X_n$ **do**
      $\mathrm{Bel}(q_i^{(n)}) \leftarrow 0$
      for $q_j^{(n-1)} = q_1^{(n-1)}, ..., q_{N_{n-1}}^{(n-1)} \in X_{n-1}$ **do**
        $s \leftarrow \frac{1}{\sqrt{2\pi \cdot \varepsilon^2}} \cdot e^{-\frac{1}{2\varepsilon^2} ||q_i^{(n)} - U_n \cdot q_j^{(n-1)}||^2}$
        $\mathrm{Bel}(q_i^{(n)}) \leftarrow \mathrm{Bel}(q_i^{(n)}) + s \cdot \mathrm{Bel}(q_j^{(n-1)})$
      end for
    end for
    $\mathrm{Normalize}(\mathrm{Bel}(q_1^{(n)}), ..., \mathrm{Bel}(q_{N_n}^{(n)}))$
      ▷ Report $\mathrm{Bel}(q_1^{(n)}), ..., \mathrm{Bel}(q_{N_n}^{(n)})$
  end for

VI. Analysis and Guarantees

In this section, we present some analysis and guarantees of our method. We first note that using the same proof of [42], our algorithm is *output sensitive*. That is, the time complexity is a function of the resulting set of possible locations for the robot.

Theorem VI.1. *Assume that each call to* $\mathrm{VoxelXPredicate}$ *and* $\mathrm{EstimateKPrime}$ *takes at most* $\mathcal{Q}_\mathcal{W}$ *time, and there are* m *reported poses at the last iteration of the algorithm. Then Algorithm 1 runs in time*

$$\Theta(k \cdot \mathcal{Q}_\mathcal{W} \cdot m \log \delta). \quad (4)$$

To further understand the magnitude of the number of reported poses, we can use the notion of *Hausdorff measure and dimension*, which is an extension of the Lebesgue measure for subsets of the Euclidean space that have Lebesgue measure zero, but still have some non-zero "area" in a lower dimension. For example, the Hausdorff dimension of a 2-manifold embedded in $\mathbb{R}^3$ is 2, and its Hausdorff measure is its area [64].

Theorem VI.2. *Assume that* M *is the intersection of all fibers corresponding to distance measurements. Then, by taking* $\delta \to 0$*, Algorithm 1 converges to* M*. Furthermore, the time complexity of our method is*

$$\Theta\Big(k \cdot \mathcal{Q}_\mathcal{W} \cdot \log \delta^{-1} \cdot (\delta_0 \cdot \delta^{-1})^{\mathcal{H}_{\dim}(M)} \cdot \mathcal{H}^{\mathcal{H}_{\dim}(M)}(M)\Big) \quad (5)$$

where δ_0 *is the diameter of* V_0*,* $\mathcal{H}_{\dim}(M)$ *is the Hausdorff dimension of* M *and* $\mathcal{H}^{\mathcal{H}_{\dim}(M)}(M)$ *is the* $\mathcal{H}_{\dim}(M)$*-dimensional Hausdorff measure of* M*.*

Proof. Proof for both theorems follows the same as [42]. □

We also note that, assuming that our estimate of k' is indeed correct, our method is robust:

Theorem VI.3. *Assuming that the* k' *estimate value of* $\mathrm{EstimateKPrime}(V)$ *is a correct lower bound to the actual number of perceived dynamic obstacles for each voxel* V*, Algorithm 1 is guaranteed to output at least one pose* q *which is* δ*-close to the ground truth location of the robot.*

An immediate corollary of Theorem VI.3 is that our method is inherently robust to kidnappings—even if the odometry is significantly off, e.g., the robot was picked up and placed in a different room, the new set of samples X_n will already contain the new location of the robot. Notably, this robustness is achieved without the need for any explicit kidnap-detection heuristics or re-sampling [65]. This is particularly useful in real-world settings, where unexpected obstacles, blocked sensors, or sudden disruptions can easily throw off the robot's pose. Our method handles these situations naturally, recovering from errors without special reset routines or manual intervention.

Finally, we note the following comparison with Monte Carlo-like methods:

Corollary VI.1. *When sampling random configurations in the bounding volume* V_0 *whose diameter is* δ_0*, to guarantee, in expectation, that we will have at least one sample that is* δ*-close to the ground truth location, we need to query*

$$O(\delta_0 \cdot \log \delta^{-1} \cdot \delta^{-\dim \mathcal{C}}) \quad (6)$$

particles.

However, if the fiber intersection M *is a set of* m *distinct poses, each contained in a ball of radius proportional to* δ*, i.e.,* $\alpha \cdot \delta$*, then our method would query only*

$$O(m \cdot V_{\dim \mathcal{C}} \cdot \alpha^{\dim \mathcal{C}} \cdot \log \delta^{-1}), \quad (7)$$

where $V_{\dim \mathcal{C}}$ *is the volume of the unit* $\dim \mathcal{C}$*-ball.*

In other words, assuming that the query time of a particle and a voxel are similar, we substitute the exponential

Fig. 4. An image of our iRobot Create 3 in a corridor of map `fl4`. The RPLIDAR A1 is mounted on top of the robot.

dependence $\delta^{-\dim\mathcal{C}}$ with $\alpha^{\dim\mathcal{C}}$, which is a significant saving for a sufficiently small α.

Proof. First, note that the upper bound $O(\delta_0 \cdot \log\delta^{-1} \cdot \delta^{-\dim\mathcal{C}})$ on random sampling is a classical result [66].

We note that the Hausdorff dimension of a collection of m $\dim\mathcal{C}$-balls is $\dim\mathcal{C}$, and their Hausdorff measure is m times the volume of each sphere, times its radius to the $\dim\mathcal{C}$-th power.

Overall, using Theorem VI.2, we get an upper bound of (we omit the $k \cdot \mathcal{Q}_{\mathcal{W}}$ as analyze the number of elements to query):

$$O\left(\log\delta^{-1} \cdot (\delta_0 \cdot \delta^{-1})^{\dim\mathcal{C}} \cdot m \cdot V_{\dim\mathcal{C}} \cdot (\alpha \cdot \delta)^{\dim\mathcal{C}}\right). \quad (8)$$

□

VII. Experiments and Results

A. Implementation Details

Our open source software[1] is an independent C++ header-only library with Python bindings in *nanobind* [44], and ROS2 [45] packages for the algorithm and for the physical robot. We use OpenMP [67] for running the voxel intersection predicate in parallel. We use CGAL AABB Tree [68] for fast axis-aligned bounding-box intersection queries.

We run our experiments on an iRobot Create 3 mobile robot (See Figure 4), equipped with a SLAMTEC RPLIDAR A1 two-dimensional LiDAR, and a Raspberry Pi 4, running ROS2 Humble. Algorithm 1 was run on a laptop with an 8-Core CPU.

B. Evaluation in Real-World Scenarios

We demonstrate our method on a physical robot in the following scenarios:

- `lab446`: Our laboratory, which is about $\sim 20[m^2]$.
- `fl4`: A full floor plan, which is about $\sim 600[m^2]$.
- `apt`: An apartment, which is about $\sim 70[m^2]$.

The maps were acquired using `slam-toolbox` [69]. All environments are dynamic in nature; in our `lab446` and in our floor `fl4`, furniture (chairs, doors) is moved, as well as people naturally walking nearby. The maps for both environments were acquired a month before the experiments. In the `apt` environment, the map was acquired a few days before the experiments. During those days, furniture was moved and rearranged slightly, due to natural, typical usage.

In each environment, the robot completes 5 laps. Each lap passes through 3 landmarks whose exact locations are pre-determined. At each lap, we measure the deviation ϵ_{pos} of position in meters, ϵ_{rot} of orientation in radians, of the approximated location of the robot on each of the pre-determined landmarks.

[1] `https://www.cgl.cs.tau.ac.il/?p=7143`

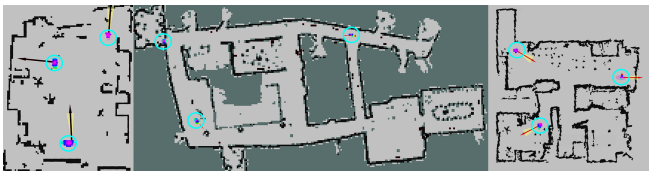

Fig. 5. Visualization of the three chosen landmarks in each environment, circled in cyan. From left to right: `fl4`, `lab446`, and `apt`. Similar to Figure 1, the colorful squares represent the pose estimate in each landmark. We overlay our pose estimate with the estimate returned by `nav2-amcl`. For clarity, all other pose candidates are omitted.

We benchmark our method against the `nav2-amcl` [70] ROS package, which is a commonly used Monte Carlo localization tool, based on Adaptive Monte Carlo Localization [65].

In Figure 5, we show the three chosen landmarks for each environment, overlaid with our belief state and the pose estimate returned by `nav2-amcl`. In the supplementary video, we showcase one lap in each environment.

In Table I we present the average errors across all scenarios, for both methods. Note that our method achieves performance comparable to state-of-the-art tools, outperforms them for some cases, and does not drift over time.

TABLE I
AVERAGE ERROR RATES ACROSS ALL SCENARIOS FOR BOTH METHODS

	lab446		**fl4**		**apt**	
	Ours	amcl	Ours	amcl	Ours	amcl
ϵ_{pos} $[m]$	0.039	**0.034**	**0.095**	0.219	0.091	**0.066**
ϵ_{rot} [rad]	**0.04**	0.159	**0.083**	0.43	**0.0247**	0.184

C. Robustness Against Kidnapped Scenarios

Recall that we solve the kidnapped robot problem from scratch at each iteration. Hence, we are able to quickly re-localize when the robot is kidnapped, without manually detecting the occurrence of such events.

For each environment, we start from a known robot location, which is precise for both our method and `nav2-amcl`. We then pick up and place the robot at 10 different locations in the environment. Furthermore, whenever possible, some furniture was moved, and at each location, there were people in the vicinity of the robot, posing as dynamic obstacles.

Immediately when the robot is placed down, we request a global re-localization from `nav2-amcl`. Our method carries on uninterrupted. For each method, we measure the time it takes to converge to a single pose. To help both methods differentiate between symmetric poses, until convergence, the robot moves slowly forward.

In `lab446`, our method converged on all ten cases immediately at the first iteration when placed down, taking less than one second. On the other hand, `nav2-amcl` converged only on 7 out of the ten placements, taking on average 59.04[sec] and at least half a minute for all cases.

In `apt`, our method still had a convergence rate of 100%, taking 12.3[sec] on average. The method `nav2-amcl` converged only on 2 out of the ten placements, taking on average 56.8[sec].

Finally, we note that on `fl4` the method `nav2-amcl` did not converge at all for any of the placements. Our method had a convergence rate of 100%, taking 15.7[sec] on average.

D. Measuring the Expected Value of k'

In Section IV-A, we discuss the need for approximating the value of k'. In lieu of imposing arbitrary assumptions on the dynamic obstacles, we collect data and learn such an approximation.

We have placed five SLAMTEC RPLIDAR S1 two-dimensional LiDARs in two environments: (i) `lab446` and the kitchenette of `fl4`, which recorded point clouds over a month and a week, respectively. The LiDARs are placed in opposing corners of each environment, so they can jointly map the entire room when there are no dynamic obstacles.

After recording, we take the union of the point clouds for each timestamp. By sampling random poses in the environment, and casting k rays in simulation, we can evaluate the value of k' that a robot placed in that random pose would perceive. We sampled for different values of $k=4,...,20$. As suggested in Section IV-A, for each sampled point we also computed its distance from the boundary of the environment $\partial\mathcal{W}$.

The processed result is a table which maps values of $\text{dist} = d((q_x,q_y),\partial\mathcal{W})$ to the k'/k ratio that would have been perceived at the time. We use the table for `lab446` to fit a decision stump [17]. We achieved the decision rule: *if* $\text{dist} < 0.975[\text{m}]$ *then the* k'/k *ratio is* 0.8116*, else* 0.7542. In Section IV-A, we round down those ratios to get different integral values when multiplying this ratio by $k=16$.

We then interdependently evaluated a decision stump for the table corresponding to the kitchenette of `fl4`, and achieved the decision rule: *if* $\text{dist} < 0.75[m]$ *then the* k'/k *ratio is* 0.8086*, else* 0.8279.

Note that the decision rules behave differently, although similar, probably due to the geometry and the different nature of the two environments. Still, we note that our suggested $\tilde{f}_{k'}(\text{dist})$ is a lower bound for both, and performed well in all scenarios tested.

VIII. Conclusions and Future Work

In this work, we demonstrate an effective method of lifelong indoor localization using merely a sparse distance sampling and the robot's odometry. While this method achieves results comparable to state-of-the-art tools, it excels in inherent robustness against significant sudden shifts in the robot's location, as we solve efficiently the kidnapped robot at every iteration, from scratch. We also consider the existence of dynamic, unforeseen changes in the pre-determined map, based on insights gathered from real-world data.

However, when the robot faces many dynamic obstacles in extreme cases, we may temporarily lose its location. Once the dynamic obstacles pass, the robot will be able to quickly re-localize, but we plan to improve upon this shortcoming in future work. We also plan to improve this localization technique further by actively controlling the robot and choosing a good intermediate motion that would allow faster convergence to the ground truth location.

Furthermore, in this work, we used a simple heuristic to learn the environment's dynamic nature. Nevertheless, more advanced machine learning tools can be incorporated to achieve even more accurate and robust results.

References

[1] A. Motroni, A. Buffi, and P. Nepa, "A Survey on Indoor Vehicle Localization Through RFID Technology," *IEEE Access*, vol. 9, pp. 17 921–17 942, 2021.

[2] G. Reina, A. Vargas, K. Nagatani, and K. Yoshida, "Adaptive Kalman Filtering for GPS-based Mobile Robot Localization," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, Sep. 2007, pp. 1–6, iSSN: 2374-3247.

[3] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of Indoor Robot Localization Techniques in the Absence of GPS," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV*, vol. 7664. SPIE, Apr. 2010, pp. 606–610.

[4] S. Yousuf and M. B. Kadri, "Sensor Fusion of INS, Odometer and GPS for Robot Localization," in *2016 IEEE Conference on Systems, Process and Control (ICSPC)*, Dec. 2016, pp. 118–123.

[5] S. Chen, D. Yin, and Y. Niu, "A Survey of Robot Swarms' Relative Localization Method," *Sensors*, vol. 22, no. 12, p. 4424, Jan. 2022.

[6] T. Yang, A. Cabani, and H. Chafouk, "A Survey of Recent Indoor Localization Scenarios and Methodologies," *Sensors*, vol. 21, no. 23, p. 8086, Jan. 2021.

[7] S.-H. Bach, P.-B. Khoi, and S.-Y. Yi, "Application of QR Code for Localization and Navigation of Indoor Mobile Robot," *IEEE Access*, vol. 11, pp. 28 384–28 390, 2023.

[8] A. Morar, A. Moldoveanu, I. Mocanu, F. Moldoveanu, I. E. Radoi, V. Asavei, A. Gradinaru, and A. Butean, "A Comprehensive Survey of Indoor Localization Methods Based on Computer Vision," *Sensors*, vol. 20, no. 9, p. 2641, Jan. 2020.

[9] P. Nazemzadeh, D. Fontanelli, D. Macii, and L. Palopoli, "Indoor Localization of Mobile Robots Through QR Code Detection and Dead Reckoning Data Fusion," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2588–2599, Dec. 2017.

[10] M. Greiff, A. Robertsson, and K. Berntorp, "Performance Bounds in Positioning with the VIVE Lighthouse System," in *2019 22th International Conference on Information Fusion (FUSION)*, Jul. 2019, pp. 1–8.

[11] M. Hoppe, M. Burger, A. Schmidt, and T. Kosch, "DronOS: a flexible open-source prototyping framework for interactive drone routines," in *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–7.

[12] A. Taffanel, B. Rousselot, J. Danielsson, K. McGuire, K. Richardsson, M. Eliasson, T. Antonsson, and W. Hönig, "Lighthouse Positioning System: Dataset, Accuracy, and Precision for UAV Research," Apr. 2021, arXiv:2104.11523 [cs].

[13] P. Debanne, J.-V. Herve, and P. Cohen, "Global Self-Localization of a Robot in Underground Mines," in *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, Oct. 1997, pp. 4400–4405 vol.5, iSSN: 1062-922X.

[14] F. Inostroza, I. Parra-Tsunekawa, and J. Ruiz-del Solar, "Robust Localization for Underground Mining Vehicles: An Application in a Room and Pillar Mine," *Sensors*, vol. 23, no. 19, p. 8059, Jan. 2023.

[15] K. Nagatani, H. Ishida, S. Yamanaka, and Y. Tanaka, "Three-dimensional Localization and Mapping for Mobile Robot in Disaster Environments," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 4, Oct. 2003, pp. 3112–3117 vol.3.

[16] D. P. Stormont and A. Kutiyanawala, "Localization Using Triangulation in Swarms of Autonomous Rescue Robots," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, Sep. 2007, pp. 1–6, iSSN: 2374-3247.

[17] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A Survey of State-of-the-art on Visual SLAM," *Expert Systems with Applications*, vol. 205, p. 117734, Nov. 2022.

[18] J. Aulinas, Y. Petillot, J. Salvi, and X. Llado, "The SLAM problem: a survey," in *Artificial Intelligence Research and Development*. IOS Press, 2008, pp. 363–371.

[19] M. U. Khan, S. A. A. Zaidi, A. Ishtiaq, S. U. R. Bukhari, S. Samer, and A. Farman, "A Comparative Survey of LiDAR-SLAM and LiDAR based Sensor Technologies," in *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, Jul. 2021, pp. 1–8.

[20] Y. Zhang, P. Shi, and J. Li, "3D LiDAR SLAM: A Survey," *The Photogrammetric Record*, vol. 39, no. 186, pp. 457–517, 2024.

[21] J. V.-V. Gerwen, K. Geebelen, J. Wan, W. Joseph, J. Hoebeke, and E. De Poorter, "Indoor Drone Positioning: Accuracy and Cost Trade-Off for Sensor Fusion," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 961–974, Jan. 2022.

[22] S. Wang, F. Colas, M. Liu, F. Mondada, and S. Magnenat, "Localization of Inexpensive Robots with Low-Bandwidth Sensors," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, and M. Gauci, Eds. Cham: Springer International Publishing, 2018, pp. 545–558.

[23] F. E. Fernandes, G. Yang, H. M. Do, and W. Sheng, "Detection of Privacy-Sensitive Situations for Social Robots in Smart Homes," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 727–732, iSSN: 2161-8089.

[24] C. Lutz and A. Tamo-Larrieux, "The Robot Privacy Paradox: Understanding How Privacy Concerns Shape Intentions to Use Social Robots," *Human-Machine Communication*, vol. 1, pp. 87–111, Jan. 2020.

[25] J. M. Pak and C. K. Ahn, "State Estimation Algorithms for Localization: A Survey," *International Journal of Control, Automation and Systems*, vol. 21, no. 9, pp. 2771–2781, Sep. 2023.

[26] Y. Fuke and E. Krotkov, "Dead reckoning for a lunar rover on uneven terrain," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, Apr. 1996, pp. 411–416 vol.1, iSSN: 1050-4729.

[27] K. Halvorsen, "Analysis of position estimation in a dead reckoning navigation robot," Master's thesis, NTNU, 2021.

[28] H. V. Do, Y. Hun Kim, J. H. Lee, M. Ho Lee, and J. W. Song, "DeRO: Dead Reckoning Based on Radar Odometry With Accelerometers Aided for Robot Localization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 8547–8554, iSSN: 2153-0866.

[29] F. Guo, H. Yang, X. Wu, H. Dong, Q. Wu, and Z. Li, "Model-Based Deep Learning for Low-Cost IMU Dead Reckoning of Wheeled Mobile Robot," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 7, pp. 7531–7541, Jul. 2024.

[30] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong Localization in Changing Environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, Dec. 2013.

[31] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary Maps for Lifelong Visual Localization," *Journal of Field Robotics*, vol. 33, no. 5, pp. 561–590, 2016.

[32] M. Zhao, X. Guo, L. Song, B. Qin, X. Shi, G. H. Lee, and G. Sun, "A General Framework for Lifelong Localization and Mapping in Changing Environment," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 3305–3312.

[33] R. B. Sousa, H. M. Sobreira, and A. P. Moreira, "A Systematic Literature Review on Long-term Localization and Mapping for Mobile Robots," *Journal of Field Robotics*, vol. 40, no. 5, pp. 1245–1322, 2023.

[34] X. Li, S. Yuan, H. Cai, S. Lu, W. Wang, and J. Liu, "LL-Localizer: A Life-Long Localization System based on Dynamic i-Octree," Apr. 2025, arXiv:2504.01583 [cs].

[35] Y. Fang, Y. Li, K. Qian, F. Tombari, Y. Wang, and G. H. Lee, "LiLoc: Lifelong Localization Using Adaptive Submap Joining and Egocentric Factor Graph," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, May 2025, pp. 8041–8047.

[36] B. Chen, X. Zhong, H. Xie, P. Peng, H. Hu, X. Zhong, and Q. Liu, "SLAM-RAMU: 3D LiDAR-IMU Lifelong SLAM with Relocalization and Autonomous Map Updating for Accurate and Reliable Navigation," *Industrial Robot: the international journal of robotics research and application*, vol. 51, no. 2, pp. 219–235, Feb. 2024.

[37] M. Mustafa, A. Stancu, N. Delanoue, and E. Codres, "Guaranteed SLAM—An interval approach," *Robotics and Autonomous Systems*, vol. 100, pp. 160–170, Feb. 2018.

[38] R. Guyonneau, S. Lagrange, L. Hardouin, and P. Lucidarme, "Guaranteed interval analysis localization for mobile robots," *Advanced Robotics*, vol. 28, no. 16, pp. 1067–1077, Aug. 2014.

[39] Y. Song, H. Yang, L. Zhao, and S. Huang, "Guaranteed 2D Pose Graph SLAM With Bounded Noises: An Efficient Interval Approach," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2025.

[40] M. M. Bilevich, S. M. LaValle, and D. Halperin, "Sensor Localization by Few Distance Measurements via the Intersection of Implicit Manifolds," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 1912–1918.

[41] M. M. Bilevich, S. Guini, and D. Halperin, "Localization in Dynamic Planar Environments Using Few Distance Measurements," Aug. 2024, arXiv:2407.03219 [cs].

[42] M. M. Bilevich, T. Buber, and D. Halperin, "Indoor Localization of UAVs Using Only Few Measurements by Output-Sensitive Preimage Intersection," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, May 2025, pp. 5773–5780.

[43] Z. Chen, "Bayesian filtering: From Kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[44] W. Jakob, "nanobind: Tiny and Efficient C++/Python Bindings," 2022.

[45] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses in the Wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[46] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. CRC Press, Oct. 2010.

[47] Z. Chen and H. Zhang, "Neural marching cubes," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 251:1–251:15, Dec. 2021.

[48] W. E. Lorensen and H. E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field, Volume 1*. New York, NY, USA: Association for Computing Machinery, Jul. 1998, vol. Volume 1, pp. 347–353.

[49] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, Oct. 2006.

[50] G. Nielson, "Dual marching cubes," in *IEEE Visualization 2004*, Oct. 2004, pp. 489–496.

[51] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter, "Meshing of Surfaces," in *Effective Computational Geometry for Curves and Surfaces*, J.-D. Boissonnat and M. Teillaud, Eds. Berlin, Heidelberg: Springer, 2006, pp. 181–229.

[52] J.-D. Boissonnat, R. Dyer, and A. Ghosh, "Delaunay stability via perturbations," *International Journal of Computational Geometry & Applications*, vol. 24, no. 02, pp. 125–152, Jun. 2014.

[53] J.-D. Boissonnat, C. Wormser, and M. Yvinec, "Anisotropic Delaunay Mesh Generation," *SIAM Journal on Computing*, vol. 44, no. 2, pp. 467–512, Jan. 2015.

[54] J. Campolattaro, S. Giraudot, C. Portaneri, T. Zhao, and P. Alliez, "Quadtrees, Octrees, and Orthtrees," in *CGAL User and Reference Manual*, 6th ed. CGAL Editorial Board, 2024. [Online]. Available: https://doc.cgal.org/6.0.1/Manual/packages.html#PkgOrthtree

[55] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, Jun. 1982.

[56] H. Samet, "An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures," in *Theoretical Foundations of Computer Graphics and CAD*, R. A. Earnshaw, Ed. Berlin, Heidelberg: Springer, 1988, pp. 51–68.

[57] C.-H. Hsu, Y.-J. Chiang, and C. Yap, "Rods and Rings: Soft Subdivision Planner for R^3 x S^2," Jun. 2019, arXiv:1903.09416 [cs].

[58] C. Wang, Y.-J. Chiang, and C. Yap, "On soft predicates in subdivision motion planning," in *Proceedings of the twenty-ninth annual symposium on Computational geometry*, ser. SoCG '13. New York, NY, USA: Association for Computing Machinery, Jun. 2013, pp. 349–358.

[59] C. K. Yap, "Soft Subdivision Search in Motion Planning, II: Axiomatics," in *Frontiers in Algorithmics*, J. Wang and C. Yap, Eds. Cham: Springer International Publishing, 2015, pp. 7–22.

[60] M. M. Bilevich and D. Halperin, "A Note on the Time Complexity of Using Subdivision Methods for the Approximation of Fibers," Mar. 2025, arXiv:2503.01626 [cs].

[61] S. M. LaValle, "Planning Algorithms," in *Planning Algorithms*. Cambridge University Press, May 2006, pp. 105–152.

[62] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, Jul. 2003.

[63] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo Localization for Mobile Robots," *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, May 2001.

[64] L. Evans, *Measure Theory and Fine Properties of Functions*. New York: Routledge, Apr. 2018.

[65] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, May 1999, pp. 1322–1328 vol.2, iSSN: 1050-4729.

[66] S. Janson, "Random coverings in several dimensions," *Acta Mathematica*, vol. 156, no. none, pp. 83–118, Jan. 1986, publisher: Institut Mittag-Leffler.

[67] OpenMP Architecture Review Board, "OpenMP Application Program Interface Version 3.0," May 2008. [Online]. Available: http://www.openmp.org/mp-documents/spec30.pdf

[68] P. Alliez, S. Tayeb, and C. Wormser, "2D and 3D Fast Intersection and Distance Computation," in *CGAL User and Reference Manual*, 6th ed. CGAL Editorial Board, 2024. [Online]. Available: https://doc.cgal.org/6.0.1/Manual/packages.html#PkgAABBTree

[69] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, May 2021.

[70] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The Marathon 2: A Navigation System," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.