Online range-based SLAM using B-spline surfaces

Rômulo T. Rodrigues[1], Nikolaos Tsiogkas[2,3], António Pascoal[4], and A. Pedro Aguiar[1]

***Abstract*— Range-based SLAM is a well-established technique for estimating the pose of a mobile robot operating in an unknown environment. Current state-of-the-art solutions use occupancy-grid maps to represent the world. While fast and accurate, their performance is limited by two facts. First, in an occupancy-grid map measurements have to be discretised into cell resolution. Second, online pose estimation, which relies on scan-to-map alignment, typically requires smoothing/interpolating the discrete grid-map. This paper presents a SLAM technique that builds on top of a B-spline surface map. The local properties of splines and the inherent smoothness of their basis function handle the aforementioned problems, without significant increase in the computational cost. Through qualitative and quantitative tests using public data sets we show that the proposed B-spline SLAM is an affordable technique that delivers accurate results at sensor rate speed.**

I. INTRODUCTION

Simultaneous Localisation and Mapping (SLAM) is a state estimation technique for concurrently estimating the pose of a mobile sensor and building a model of its surrounding environment. It has been widely adopted for deploying sensor-controlled robots whenever an external referencing system works poorly or is not available. Since SLAM provides online state feedback to a number of tasks, including motion planning and motion control, a poor SLAM solution may lead to undesired behaviours, such as mission failure or equipment damage.

Over the last decades, the SLAM problem has received considerable attention. Different techniques have been successfully demonstrated to work, e.g., [1], [2], [3]. Robustness has been achieved by relying on a two stage architecture, namely, the *front-end* and *back-end* stages [4]. The front-end stage, also known as online-SLAM, is responsible for processing the raw sensor data and providing state estimation at least as fast as the sensor rate. Then, at lower rates, the back-end optimises the overall state estimation using graph-based optimisation techniques. The nodes of the graph to be optimised are the estimated pose of the robot and the edges are motion constraints that relate two poses, such as the odometry of the robot. The front-end provides both pose estimation and motion constraints for building the graph. Sources such as wheel encoders and *Inertial Measurements Units* (IMU) potentially enrich the graph informativeness leading to better results. This paper focuses on the front-end of a 2D range-based SLAM.

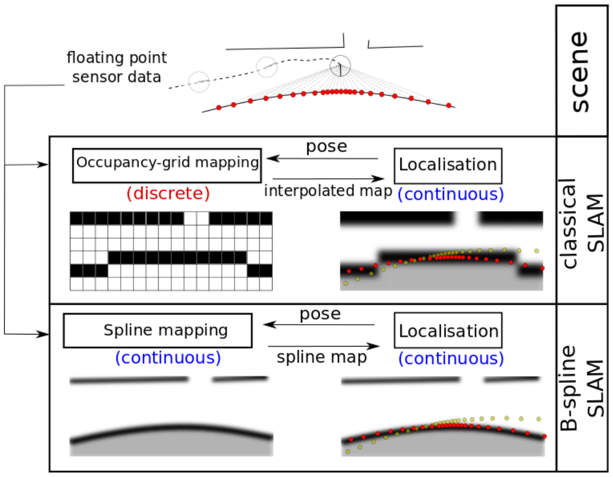

Fig. 1. Comparing classical (occupancy grid) SLAM and B-spline SLAM. The top of the image shows a mobile robot equipped with a range sensor. Measurements are shown in yellow (before scan-matching) and red dots (after scan-matching). The middle image illustrates the discrete map which is stored in occupancy-grid SLAM. Typically, for localisation, a smoothed/continuous version of regions of the map is derived for computing scan-matching using gradient-based strategies. The bottom image shows the proposed B-spline SLAM strategy. The map is stored in a B-spline surface and localisation operates directly on it. The B-spline surface map is less impacted by the misalignment between cells and scan hits, and potentially yields better scan-to-map alignment results.

The front-end of current state-of-the-art range-based SLAM algorithms runs a scan matching routine, which searches for the most likely pose that matches the current reading against the map in memory (see Fig. 1). Also called scan-to-map alignment, it can be solved using brute-force approaches, e.g., [5]. However, in recent SLAM strategies, such as Google Cartographer [3] and Hector-SLAM [2], scan matching is formulated as an optimisation problem and solved using gradient descent methods. Performing the optimisation efficiently requires a continuous map. Thus, when storing the map in a discrete grid, interpolation techniques come into play, e.g., non-smooth linear approximation [2] or bi-cubic interpolation [3]. In a nutshell, occupancy grid-based SLAM converts floating point measurements into discrete resolution for efficient storage. Later, scan-matching

This work was partly supported by Doctoral program PDMA-NORTE-08-5369-FSE-000061; FCT R&D Unit SYSTEC UIDB-P/00147/2020; IMPROVE POCI-01-0145-FEDER-031823; RELIABLE (PTDC/EEI-AUT/3522/2020); HARMONY POCI-01-0145-FEDER-031411; MAGIC POCI-01-0145-FEDER-032485; Flanders Make/VLAIO: SBO MULTIROB; H2020 EUMR Infrastructures Network (Project ID 731103) and the FCT Project UID/EEA/5009/2013.

[1]Rômulo T. Rodrigues and A. Pedro Aguiar are with Faculty of Engineering, University of Porto, Portugal {romulortr,pedro.aguiar}@fe.up.pt

[2]Nikolaos Tsiogkas is with Department of Mechanical Engineering, Division RAM, KU Leuven, Leuven, Belgium and with

[3]FlandersMake@KULeuven, Core Lab ROB, Leuven, Belgium nikolaos.tsiogkas@kuleuven.be

[4]António Pascoal is with Laboratory of Robotics and Engineering Systems (LARSyS), ISR/IST, University of Lisbon, Lisbon, Portugal antonio@isr.tecnico.ulisboa.pt

requires the computation of derivatives and sub-cell accuracy which is achieved by interpolating the discrete grid. We believe that such a process leads to loss of information and potentially degraded performance.

Continuous metric maps that use geometric features such as lines [6] and curves [7] are able to encode in their storage format the floating point nature of the data. However, their popularity is still quite limited within the SLAM community. From our previous experience with a geometric map-based SLAM [8], we believe that possible bottlenecks for its acceptance are that merging geometric primitives is challenging and checking whether a region is free or occupied - an important query in motion planning algorithms - can be troublesome.

In this paper we propose an online B-spline SLAM framework that builds on top of the continuous B-spline surface map presented in [9]. Figure 1 illustrates the main advantages of a B-spline map over a discrete occupancy-grid map for the SLAM problem: 1) the impact of an occupied measurement is greatest at the corresponding point (not cell) of the map, and 2) smoothness is guaranteed by the inherent properties of splines. No interpolation is required because the map itself is continuous. The computational cost for updating/querying the B-spline map is independent of the size of the map and proportional to the order of the spline squared (constant). Our results show that a low budget laptop is able to run public data sets at sensor frame rate.

In our previous work [9] we presented a B-spline mapping framework assuming that the pose of the sensor is known. The main contributions of this paper are:

1) **B-spline SLAM**: The core contribution is the novel online B-spline SLAM algorithm described in Sec. IV and its comparison with other relevant SLAM algorithms using public data sets, showing improved results.
2) **Open-source code**: The theoretical results presented in this paper are backed up by a publicly available package with examples, including ROS integration [1].

The rest of the paper is organised as follows. In Sec. II, we discuss previous work on the SLAM problem. Sec. III introduces B-splines, the main tool employed in our solution. In Sec. IV, we present the theoretical development of the proposed strategy. Then, in Sec. V, we assess the performance of B-spline SLAM using public data sets and comparing with other relevant SLAM strategies. Finally, Sec. VI concludes the paper.

II. LITERATURE REVIEW

The state in SLAM encodes the map and the pose of the robot. Estimating the map is known as mapping, while estimating the pose of the robot is named localisation. The literature review presented here focuses mainly on front-end SLAM, from classical and well-established solutions to more recent approaches that have shown promising results.

[1]https://github.com/C2SR/spline-slam

A. Landmarks

Landmark maps store the location of artificial or non-artificial features of the environment. The landmark based EKF-SLAM was a pioneering solution proposed by Smith et al. [10]. It keeps track of the posterior distribution of the SLAM state using an Extended Kalman Filter (EKF). The main advantages of EKF-SLAM is that it is simple to implement and the results are good if the Gaussian model assumptions are not substantially violated. The main drawbacks are that 1) the computational effort to incorporate an observation grows quadratically in the number of landmarks and 2) the method requires data association. Dissanayake and colleagues [11] proposed removing landmarks based on the information content. This improved the computational efficiency of EKF-SLAM by keeping the filter compact.

Murphy [12] was one the first authors to apply Rao-Blackwellized particle filter (RBPF) to the solution of the SLAM problem. Few years later, Montemerlo et al. presented FastSLAM [13], an online strategy able to deal with thousands of landmarks, something that was impossible with EKF-SLAM approaches. In FastSLAM, a particle (or sample) represents a hypothesis on the path of the robot - leading to the term multi-belief or multi-hypotheses filter. For mapping, each particle keeps track of the position of the landmarks. For path estimation, particles are propagated in a process called sampling from the proposal distribution. Each particle has an importance weight that indicates the likelihood of the observations reported by the sensor. Based on these weights, a resampling policy decides which particles will survive or be removed. The key advantages of RBPF are that it 1) relaxes the Gaussian assumption, 2) deals better with non-linearities, and 3) is computationally more efficient than previous approaches. The main drawback is the particle depletion problem: during the resampling, particles that are actually good may be eliminated. FastSLAM 2.0 [14] addressed the particle depletion problem by taking into account the observations from the measurements for computing an improved proposal distribution.

B. Occupancy grid

Occupancy grid maps have been around since 1985, when Moravec and Elfes [15] proposed them as method for registering dense range measurements. Its usage in SLAM, was enabled by the emergence of scan-to-scan alignment methods such as the Iterative Dual Correspondence, a two stage search method, originally presented by Lu and Milios [16]. The first search minimises the translation error by matching the points that are the closest and minimising their distance (similar to the Iterative Closest Point algorithm by Besl and McKay [17]). The second step refines the previous estimate by matching range points that are within the same distance in their respective reference frame. Hähnel et al. [18] presented a scan-to-map alignment method inspired in IDC that is able to detect, track, and filter out people in the vicinity of the robot. Locally, scan-to-map alignment presented good results, but small drifts make it hard to deal with large loop closure. Degradation of the map quickly deteriorates localisation and vice-versa. To overcome that multi-belief filters and offline strategies have been employed.

Hähnel et al. [19] presented an online grid-based SLAM using a RBPF. A scan-matching routine transforms a fixed window of consecutive range measurements into accurate odometry measurements. The range-based odometry has lower variance than pure wheel encoder odometry and it is employed for generating a more accurate proposal distribution for the particles. Grisetti et al. [20] proposed running a scan matching process for each particle, instead of having a fixed proposal distribution for all the particles as in [19]. While by doing so their strategy improved the proposal distribution for an individual particle, the odometry information was not properly considered. The same authors proposed the GMapping algorithm in [1], an efficient, mature, and popular grid-based RBPF that it is still widely employed by the robotic community. It is an extension of [20] that considers the odometry information. The enhanced proposal distribution draws new particles more accurately. GMapping also uses an adaptive resampling technique that estimates the performance of a set of particles, which reduces the number of required particles without compromising the solution

The popularity of scan-to-map alignment increased with the advent of modern LiDAR with large angular field of view and fast frame rate. Kohlbrecher et al. [2] presented Hector-SLAM, a fast single-belief strategy based on scan-to-map alignment that uses multiple resolution grid maps. Continuous maps and their gradients are obtained by bilinear filtering. The method provides accurate estimation when the sensor rate is high enough. More recently, Hess and colleagues proposed Google Cartographer [3]. This method explicitly detects loop closure through a branch-and-bound approach and performs offline pose optimisation, the front-end is very similar to Hector-SLAM. However, continuous maps are obtained using bi-cubic interpolation.

C. More recent approaches

Range based SLAM is an active research topic. Recent work has aimed at improving SLAM performance by tackling different fronts, such as the front-end [21], the back-end [22], or both [23], [24].

Zhao et al. [21] proposed a feature-based SLAM using implicit functions, which allows representing a wide variety of shapes. The authors formulate SLAM as an energy minimisation problem, where the optimisation vector describes the pose of the robot and the changeable parameters of the implicit functions. They show that using implicit functions to represent geometric features outperforms using pre-fixed geometric shapes such as lines and ellipses. Xiong et al. [22] presented a two-level optimisation approach. Local EKF-SLAM filters keep track of sub-maps, which treat raw laser scans as landmarks. When a sub-map is completed, it triggers a loop closure detection process and updates a global graph. A two-level optimisation process optimises the global graph and refines local sub-maps. Ren and colleagues [23] proposed a correlative scan matching approach to improve front-end scan matching in dynamic environments using occupancy-grid maps. In addition to this, the authors proposed a back-end that is more robust against false loop closure detections. Daun et al. [24] use truncated signed distance functions (TSDFs) instead of a discrete occupancy grid. The cells in TSDFs describe the distance to the nearest obstacle. This provides sub-pixel precision and increases the region of convergence of gradient-descent methods during localisation. A back-end is responsible for loop closure detection and pose graph optimisation.

The proposed online B-spline SLAM is inspired by lightweight single-belief pure front-end approaches like Hector-SLAM[2]. However, instead of using a discrete grid map, we use the B-spline surface map presented in [9]. Like TSDFs, B-spline maps describe the environment more accurately than discrete grid maps and provide a larger basin of convergence during localisation. The results show that we can compete with some of the strategies discussed in this Section, namely GMapping [1], Hector-SLAM [2], Cartographer [3], two-level optimisation [22], and TSDF [24].

III. Preliminaries

A. Notation

The following notation is adopted. Scalar values are written in lower-case letters and vectors in lower-case bold letters. Matrix and random variables are typed in upper-case letters. Given a random variable X with probability distribution $p(X)$, the probability of $X = x$ is shortened as $p(x)$. Throughout the text, the words spline and B-spline are used interchangeably.

B. B-spline function

A B-spline is a vector-valued function $\mathbf{b}(\tau) : \mathbb{R} \to \mathbb{R}^m$ that spans a polynomial space of degree d and order $d+1$. The knot vector $\{t_i\}_{i=0}^{m+d}$, with $t_i \leq t_{i+1}, \forall i$, is said to support the spline function. Let $b_i^d(\tau)$ be the i-th coefficient of $\mathbf{b}(\tau)$, it follows from the De Boor's recursive algorithm [25] that

$$b_i^r(\tau) = \frac{\tau - t_i}{t_{i+r} - t_i} b_i^{r-1}(\tau) + \frac{t_{i+r+1} - \tau}{t_{i+r+1} - t_{i+1}} b_{i+1}^{r-1}(\tau), \quad (1)$$

where, in particular,

$$b_i^0(\tau) = \begin{cases} 1 & , t_i \leq \tau < t_{i+1} \\ 0 & , \text{otherwise} \end{cases}. \quad (2)$$

From the definition of B-splines stated in (1) and (2), the following properties hold:

Property 1: (*Local support*) For $\tau \in [t_\mu, t_{\mu+1})$ the function $\mathbf{b}(\tau)$ has at most $d+1$ non-zeros coefficients. These are the coefficients $b_{\mu-d}^d, \ldots, b_\mu^d$.

Property 2: (*Continuity*) Suppose that the knot t_μ occurs κ times among the knots $(t_i)_{i=\mu-d}^{\mu+d}$, with κ some integer bounded by $1 \leq \kappa \leq d+1$. Then, the spline function $\mathbf{b}(\tau)$ has continuous derivatives up to order $d - \kappa$ at the knot t_μ.

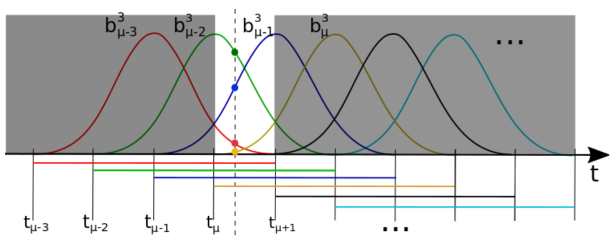

Fig. 2. A cubic B-spline function. Colored circles indicate the non-zero coefficients of the spline function at the evaluation point specified by the vertical dashed line. For a cubic spline, only 4 coefficients are non-zero. This is called the *local support* and it widely employed in B-spline SLAM for speeding up computations.

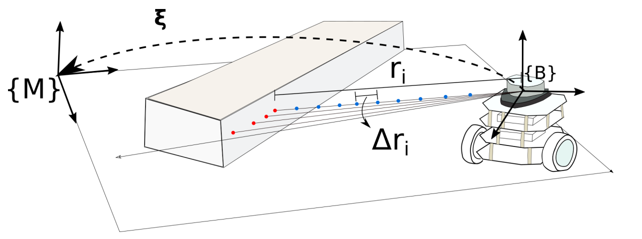

Fig. 3. Problem setup: a robot equipped with a range sensor detects occupied space (red dots) at discrete intervals. The points between the robot and the obstacle are assumed as free space (blue dots) and obtained at sampling intervals of Δr along a beam. The pose of the robot $\boldsymbol{\xi}$ is unknown and has to be estimated online.

The local support property is shown in Fig. 2 for $d = 3$. For a cubic B-spline function only four coefficients are non-zero. Note that typically $d \ll m$, so instead of computing m coefficients, only $d+1$ have to be considered, increasing considerable the computational efficiency. The SLAM framework presented here uses uniformly spaced non-clamped knot vectors, which means that the interval between any two consecutive knots is the same and the initial/final knots are not repeated. Thus, from the continuity property, we have that our spline function has derivatives up to order $d-1$.

C. B-spline surface

A B-spline surface is a scalar-valued function $\mathbf{s}(\boldsymbol{\tau}) : \mathbb{R}^2 \to \mathbb{R}$, with $\boldsymbol{\tau} = [\tau_x, \tau_y]^T \in \mathbb{R}^2$, defined by the tensor product of two B-spline functions and the control points $\{\mathbf{c}_{ij}\}$ as follows:

$$\mathbf{s}(\boldsymbol{\tau}) = \sum_{i=0}^{m_x-1} \sum_{j=0}^{m_y-1} \mathbf{c}_{ij} b_i^{d_x}(\tau_x) b_j^{d_y}(\tau_y),$$

where d_x and d_y are the degrees of the spline functions $\mathbf{b}_x(\tau_x)$ and $\mathbf{b}_y(\tau_y)$, respectively.

Let $C \in \mathbb{R}^{m_x \times m_y}$ be a real matrix with entries $\mathbf{c}_{ij}$. Then, a B-spline surface can be written in matrix form as

$$\mathbf{s}(\boldsymbol{\tau}) = \mathbf{b}_x(\tau_x)^T C \mathbf{b}_y(\tau_y). \tag{3}$$

Now, define *vec* as the vectorization operator - a linear transformation that stacks the columns of a matrix on top of each other, yielding a single column-vector. Applying the vectorization operator in (3):

$$\begin{aligned} s(\boldsymbol{\tau}) &= \operatorname{vec}\left(\mathbf{b}_x(\tau_x)^T C \mathbf{b}_y(\tau_y)\right) \\ &= \operatorname{vec}\left(\mathbf{b}_y(\tau_y)^T \otimes \mathbf{b}_x(\tau_x)^T\right) \operatorname{vec}(C) \\ &= \boldsymbol{\phi}(\boldsymbol{\tau})^T \mathbf{c}, \end{aligned} \tag{4}$$

where $\boldsymbol{\phi}(\boldsymbol{\tau})^T = \operatorname{vec}\left(\mathbf{b}_y(\tau_y)^T \otimes \mathbf{b}_x(\tau_x)^T\right)$, $\mathbf{c} = \operatorname{vec}(C)$, and $\otimes$ stands for the Kronecker product.

For a uniformly spaced knot vector, the spline function is continuous up to degree d (see Property 2). In this case, the first derivative of the surface with respect to the parametric variable exists and can be written in a compact form as:

$$\frac{ds(\boldsymbol{\tau})}{d\boldsymbol{\tau}} = \frac{d\boldsymbol{\phi}(\boldsymbol{\tau})}{d\boldsymbol{\tau}}^T \mathbf{c} = \left[\boldsymbol{\phi}_x(\boldsymbol{\tau})^T \quad \boldsymbol{\phi}_y(\boldsymbol{\tau})^T\right]^T \mathbf{c} \tag{5}$$

where the partial derivatives of the spline tensor are defined as: $\boldsymbol{\phi}_x(\boldsymbol{\tau})^T = \operatorname{vec}\left(\mathbf{b}_y(\tau_y)^T \otimes \frac{d\mathbf{b}_x(\tau_x)}{d\tau_x}^T\right)$ and $\boldsymbol{\phi}_y(\boldsymbol{\tau})^T = \operatorname{vec}\left(\frac{d\mathbf{b}_y(\tau_y)}{d\tau_y}^T \otimes \mathbf{b}_x(\tau_x)^T\right)$.

The control points define the shape of the B-spline surface, while the knot vectors define the resolution and the rate of change of the surface. We modify the map by updating the control points. For higher resolution maps, we decrease the knot interval. The final property introduced is the convex hull property [26].

Property 3 (Convex hull): A B-spline surface $\mathbf{s}(\boldsymbol{\tau})$ lies within the convex hull of its control points ($\mathbf{c}$). That results form the fact that the B-spline coefficients are non-negative and always add up to 1, defining a convex combination.

IV. B-SPLINE SURFACE SLAM

Consider an inertial coordinate frame $\{M\}$ attached to the origin of the map and a body fixed coordinate frame $\{B\}$ attached to the center of mass of a vehicle equipped with a 2D range sensor, as shown in Fig. 3. For the sake of simplicity, assume that the sensor lies at the center of mass of the vehicle. The sensor provides l range measurements of the environment $(r_i)_{i=0}^{l-1}$ at discrete angle intervals $(\alpha_i)_{i=0}^{l-1}$ w.r.t. the x-axis of $\{B\}$. Applying polar to cartesian transformation, we obtain the coordinates of the end point of the beams:

$${}^B\boldsymbol{\tau}_i^{occ} = r_i \begin{bmatrix} \cos\alpha_i \\ \sin\alpha_i \end{bmatrix}, \quad \forall i = 0, \ldots, l-1. \tag{6}$$

The superscript *occ* indicates that these measurements correspond to occupied space. Equivalently, let ${}^B\boldsymbol{\tau}_{i,j}^{free}$ be the discrete samples virtually detected as free space, i.e., not having an obstacle. For this purpose, we take samples between the end point of each beam and the robot:

$$\begin{aligned} {}^B\boldsymbol{\tau}_{i,j}^{free} = n\Delta r \begin{bmatrix} \cos\alpha_i \\ \sin\alpha_i \end{bmatrix}, &\forall i = 0, \ldots, l-1, \\ & j = 0, \ldots, r_i/\Delta r - 1, \end{aligned} \tag{7}$$

where Δr is an appropriate sampling interval. In total, we assume that there are m free space virtual measurements. We call them virtual measurements because free space is not implicitly detected by the sensor.

The relationship between two coordinate frames is encoded by $\boldsymbol{\xi} = [x, y, \psi]^T$. A vector ${}^B\boldsymbol{\tau}_i$ described in the body frame is transformed to the map frame through the transformation function T:

$$T(\boldsymbol{\xi}, {}^{B}\boldsymbol{\tau}_i) = R(\psi)^{B}\boldsymbol{\tau}_i + \begin{bmatrix} x \\ y \end{bmatrix}, \text{ with } R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{8}$$

where $R(\psi)$ is a rotation matrix.

This remainder section presents the theoretical development of the SLAM strategy proposed here. First, in Sec. IV-A we address the mapping task. Then, the localization method is discussed in Sec. IV-B.

A. Mapping

Traditionally, spline regression employs ordinary least squares [26]. It requires solving a linear system, which it is known to have a computational cost of $O(n^3)$ using the LU or QR decomposition, where n is the number of samples. A method with such complexity is inappropriate for solving the SLAM problem, where data is continuously acquired at high rates and must be processed online. In [9], we proposed an approximation using a recursive scheme, which decreases the computational cost for mapping application to $O(n)$. A brief introduction to the B-spline mapping is presented here for the sake of completeness. The interested reader is referred to the aforementioned paper for details.

Let M_i and Z_i be discrete random variables that represent the occupancy state and the sensor perception at $\boldsymbol{\tau}_i = [\tau_x, \tau_y]$, respectively. The occupancy state is either free ($M_i = 0$) or occupied ($M_i = 1$). Similarly, the report of the sensor can be either free ($Z_i = 0$) or occupied ($Z_i = 1$). In probabilistic mapping, the goal is to keep track of the probability of a space being occupied or free, i.e., $p(M_i = 1)$ and $p(M_i = 0)$. Using Bayes theorem and $\log \text{odds}$ probabilities, we can obtain the well-known recursive grid cell update:

$$\log\text{odds}\,(\hat{m}_i) = \log\text{odds}\,(\check{m}_i) + \kappa, \tag{9}$$

where $\log\text{odds}\,(\hat{m}_i)$ is the posterior, $\log\text{odds}\,(\check{m}_i)$ is the prior, and κ is a constant map update factor that takes a positive value for occupied space measurement and a negative value for free space measurement. The B-spline map approximates (9) as

$$\log\text{odds}(m_i) \approx s(\boldsymbol{\tau}_i) = \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}, \tag{10}$$

where $\mathbf{c}$ are the control points that define the spline surface map. The control points are updated when new measurements arrive as

$$\hat{\mathbf{c}} = \check{\mathbf{c}} + \frac{\boldsymbol{\phi}(\boldsymbol{\tau}_i)}{\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa,$$

To avoid an unbounded control points growth they are saturated:

$$\hat{\mathbf{c}} = \min(\max\,(\hat{\mathbf{c}}, c_{\min}), c_{\max}). \tag{11}$$

A similar clamping technique is described in [3]. From Property 3, it follows that the B-spline surface is also within the limits defined by $c_{\min}$ and $c_{\max}$.

B. Localization

The localization stack computes an estimate of the pose of the robot via scan-to-map alignment. For that, we define the cost function[2]

$$J(\hat{\boldsymbol{\xi}}) = \sum_{i=0}^{n-1}[1 - s(\boldsymbol{\tau}_i(\hat{\boldsymbol{\xi}}))]^2,$$

where $\boldsymbol{\tau}_i(\hat{\boldsymbol{\xi}}) = T(\hat{\boldsymbol{\xi}}, {}^{B}\boldsymbol{\tau}_i^{occ})$ is the estimated map coordinate of the endpoint of a beam, i.e., occupied space. The scan-to-map cost function describes how well a pose estimate aligns the current scan measurements and the map. Given an initial guess $\check{\boldsymbol{\xi}}$, we decompose the pose as $\hat{\boldsymbol{\xi}} = \check{\boldsymbol{\xi}} + \Delta\boldsymbol{\xi}$ and formulate the localization problem as

$$\min_{\Delta\xi} \sum_{i=0}^{n}[1 - \mathbf{s}(\boldsymbol{\tau}_i(\check{\boldsymbol{\xi}} + \Delta\boldsymbol{\xi}))]^2. \tag{12}$$

For solving this non-linear least square problem, we assume that the dynamic of the sensor is fast enough such that the displacement $\Delta\boldsymbol{\xi}$ is small. In this case, the non-linear function can be approximated using a Taylor expansion around $\Delta\boldsymbol{\xi} = \mathbf{0}$, that is,

$$s(\boldsymbol{\tau}_i(\check{\boldsymbol{\xi}} + \Delta\boldsymbol{\xi})) \approx s(\boldsymbol{\tau}_i(\check{\boldsymbol{\xi}})) + \left.\frac{\partial s(\boldsymbol{\tau}_i(\boldsymbol{\xi}))}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\check{\boldsymbol{\xi}}} \Delta\boldsymbol{\xi}. \tag{13}$$

Define the variables e_i and $\mathbf{h}_i$ as

$$e_i = 1 - s(\boldsymbol{\tau}_i(\check{\boldsymbol{\xi}})), \tag{14}$$

$$\mathbf{h}_i^T = \left.\frac{\partial s(\boldsymbol{\tau}_i(\boldsymbol{\xi}))}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\check{\boldsymbol{\xi}}}. \tag{15}$$

Approximating the non-linear function in (12) by (13) and substituting (14) and (15), yields

$$\min_{\Delta\xi} \sum_{i=0}^{n-1}[e_i - \mathbf{h}_i^T \Delta\boldsymbol{\xi}]^2. \tag{16}$$

The least squares solution can be obtained by derivating the cost function in (16) with respect to $\Delta\boldsymbol{\xi}$ and equating to zero, yielding after some algebraic manipulations

$$\Delta\boldsymbol{\xi} = -(\sum_{i=0}^{n-1}\mathbf{h}_i\mathbf{h}_i^T)^{-1}\sum_{i=0}^{n-1}\mathbf{h}_i e_i. \tag{17}$$

For the sake of completeness, we derive the term $\mathbf{h}_i$. From the definition of a spline surface in (4), its derivative w.r.t. $\boldsymbol{\xi}$ is

$$\mathbf{h}_i = \mathbf{c}^T \left.\frac{\partial \boldsymbol{\phi}(\boldsymbol{\tau}_i(\boldsymbol{\xi}))}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\check{\boldsymbol{\xi}}} = \mathbf{c}^T \left.\left[\frac{\partial \boldsymbol{\phi}(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}_i}\frac{\partial \boldsymbol{\tau}_i}{\partial \boldsymbol{\xi}}\right]\right|_{\boldsymbol{\xi}=\check{\boldsymbol{\xi}}}, \tag{18}$$

where $\frac{\partial \boldsymbol{\phi}(\boldsymbol{\tau}_i)}{\partial \boldsymbol{\tau}_i}$ is as defined in (5) and

$$\frac{\partial \boldsymbol{\tau}_i}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 1 & 0 & -{}^{B}\tau_{i,x}\sin\theta - {}^{B}\tau_{i,y}\cos\theta \\ 0 & 1 & {}^{B}\tau_{i,x}\cos\theta - {}^{B}\tau_{i,y}\sin\theta \end{bmatrix}. \tag{19}$$

[2]For simplicity of the presentation, we adopted a quadratic loss function. The proposed approach supports other functions like the robust loss functions Cauchy, German-McClure, and Welsch [27].

The B-spline SLAM algorithm is described in Algorithm 1. When solving the optimization problem described in (12) via the Gauss-Newton method, we use an adaptive step λ. The stop criterion are the maximum number of iterations and ΔJ_{tol}. The former parameter ensures that the solver finishes in an online-acceptable time. The latter parameter describes the maximum value required in improving the objective function before accepting a solution. Moreover, we use a multi-map resolution approach, similar to [2]. The resolution of a B-spline map is given by the knot interval: the smaller the interval, the higher is the resolution. A solution to the scan-matching problem is first computed for the lowest map resolution. Then, the obtained solution is used as a "hot start" for the next map resolution. We follow these steps from the lowest to the highest resolution. This increases the robustness of the localization stack against local minima.

Algorithm 1: B-spline SLAM algorithm

```
input      : Prior pose $\check{\boldsymbol{\xi}}$, prior map $\check{\mathbf{c}}$, range scans $(r_i)_{i=0}^{l-1}$
output     : Posterior pose $\hat{\boldsymbol{\xi}}$, posterior map $\hat{\mathbf{c}}$
initialize : $\Delta J \leftarrow \infty$, #iterations $\leftarrow 0$, $\lambda \leftarrow 1$
parameters: $\Delta J_{tol}$, max_iterations, $\kappa$

1: Remove spurious measurements
2: Range scans to Cartesian coord.: $\{{}^B\boldsymbol{\tau}_i^{occ}\}_{i=1}^n \leftarrow$ (6)
// Localization
for each map resolution (low to high) do
    while $\Delta J > \Delta J_{tol}$ and #iterations < max_iterations do
        3: Local to global frame via $\check{\boldsymbol{\xi}}$: $\{\boldsymbol{\tau}_i^{occ}\}_{i=1}^n \leftarrow$ (8)
        4: Scan-to-map alignment error: $\{e_i\}_{i=1}^n \leftarrow$(14)
        5: Jacobian: $\{h_i\}_{i=1}^n \leftarrow$ (18),(19),(5)
        6: Pose update: $\Delta\boldsymbol{\xi} \leftarrow$ (17)
        7: Cost improvement: $\Delta J \leftarrow J(\check{\boldsymbol{\xi}} + \Delta\boldsymbol{\xi}) - J(\check{\boldsymbol{\xi}})$
        if $\Delta J < 0$ then
            8.1: $\check{\boldsymbol{\xi}} \leftarrow \check{\boldsymbol{\xi}} + \lambda\Delta\boldsymbol{\xi}$
            8.2: $\lambda \leftarrow 1.5\lambda$
        else
            8.3: $\lambda \leftarrow .5\lambda$
        9: #iterations $\leftarrow$ #iterations + 1
10: Update posterior: $\hat{\boldsymbol{\xi}} \leftarrow \check{\boldsymbol{\xi}}$
// Mapping
for each map resolution do
    11: Detect free space: $\{{}^B\boldsymbol{\tau}_{i,j}^{free}\}_{i=1}^m \leftarrow$(7)
    12: Transform free and occupied space to global coord. frame using $\hat{\boldsymbol{\xi}}$: $\{\boldsymbol{\tau}_i^{occ}\}_{i=1}^n, \{\boldsymbol{\tau}_i^{free}\}_{i=1}^m \leftarrow$ (8)
    13: Update the B-spline map: $\hat{\mathbf{c}} \leftarrow$(11)
    14: Clamp control points: $\hat{\mathbf{c}} \leftarrow$(10)
```

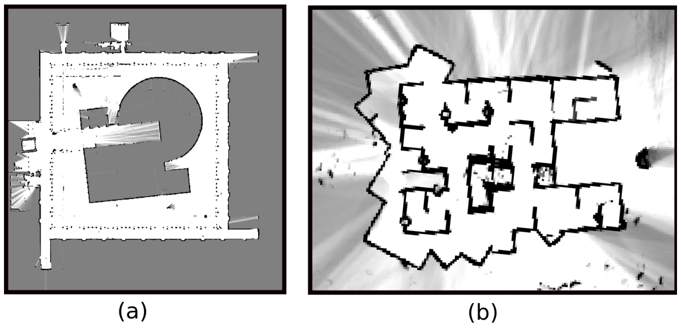

Fig. 4. B-spline SLAM map for TU Darmstadt dataset: (a) Dagstuhl and (b) RoboCup 2010.

V. EXPERIMENTS

The proposed algorithm was evaluated using public data sets that provide different sensors and odometry measurements. In our tests, the main characteristic that has an impact on our algorithm is the sensor rate. The higher the rate of the sensor, the better the assumption that the pose displacement between two readings is small holds, i.e., $\Delta\boldsymbol{\xi} \approx \mathbf{0}$. Therefore, when using low rate sensors more map resolutions are required to avoid getting trapped in a local minimum during localisation. The parameters of the proposed algorithm are the same for all experiments, except for the number of map resolutions, which is clearly stated. For visualisation, the B-spline surface is sampled at 0.05 m interval and displayed as a gray-scale image. The darker a pixel is, the more likely it is to correspond to an occupied area.

A. TU Darmstadt data set

(Map resolutions: 1, knot interval: 5 cm) The TU Darmstadt data set contains data recorded using an IMU and a Hokuyo UTM-30LX LIDAR (45 Hz). The resulting map for two scenarios computed by B-spline SLAM are shown in Fig. 4. The two sensors (IMU and LiDAR) were assembled in a handheld kit. For instance, the data from the Dagstuhl building (Fig. 4a) was acquired by a human walking through the environment carrying the sensors by hand. The sensor was subject to roll, pitch, and vertical oscillations breaking the planar movement constraint. The results presented in [2] (Hector-SLAM) correct the measurements by incorporating the IMU data to obtain a stabilised coordinate frame. In our SLAM framework we do not take the IMU data into account. Since no ground truth is provided, it is hard to assert whether a mapping structure presented in our solution but not in [2] represents an advantage or disadvantage. Given that both maps in Fig. 4 are coherent, we conclude that our results are good in a qualitative sense.

B. Radish data set

(Map resolutions: 3, knot interval: 5, 12.5, 30 cm) The Robotics Data Set Repository (Radish) [28] contains odometry data and range scan measurements ($\sim$5 Hz). Figure 5 shows the qualitative results using the proposed algorithm in four of the five scenarios evaluated. For quantitative results, we use the metric proposed by Kummerle et al. [29]. The

TABLE I

Main features of compared methods.

	Map	Belief	Loop-closure	Back-end
B-spline SLAM	B-spline	single	no	no
RBPF	discrete grid	multi	yes[1]	no
Cartographer	discrete grid	single	yes	yes
Two-level	landmarks	single	yes	yes
TSDF	TSDF	single	yes	yes

[1]Particles that do not explain the loop closure are likely to be removed.

authors show that using global pose (i.e., a fixed reference frame) is sub-optimal for comparing SLAM algorithms. Instead, they propose comparing the translational and rotational errors between two relative poses (absolute and squared errors). For the Radish data set, a fair ground truth for several relative poses were obtained by human operators with knowledge of the building [29].

Tables I and II show the comparison of our algorithm with others in the literature. Table I describes the properties inherent to each strategy. Table II quotes the results for RBPF with 50 particles (GMapping) [29], Cartographer [3], two-level optimisation [22], and TSDF [24]. In bold, we highlight the best results and a dash indicates no information was provided by the authors for the corresponding data set. The methods being compared use the odometry data available in the log files. We tried to obtain results using Hector-SLAM but unfortunately it was failing to produce a correct map despite extensive tuning efforts. It is hypothesized that the high angular displacement between consecutive readings did not allow the correct operation of that SLAM method. On the other hand, B-spline SLAM fails only in the MIT Killian Court due to the long and narrow corridors which result to the infinite corridor problem. Other than this, it performs well. In the RBPF (50 part) algorithm there are 50 particles, and for each particle a scan matching on a discrete occupancy grid map is performed. The fact that we are consistently better than RBPF shows the potential of B-spline based SLAM. Comparing with more recent methods that run back-end optimisation, the proposed strategy is still competitive: it outperforms Cartographer for the MIT CSAIL data set and it is able to achieve either the smallest absolute or squared rotational error in the ACES, Intel, and Freiburg bld 69 data sets. It is likely that the accuracy of the proposed SLAM comes from the fact that the continuous B-spline maps are more accurate than discrete maps (see [9]) and allow for a larger basin of convergence when performing scan-matching. There is still room to improve B-spline SLAM, since in contrast to these other methods we do not perform any loop-closure mechanism or back-end optimisation.

The implementation of the proposed algorithm can process the Radish data set 1.5 to 2 times faster than the sensor rate (evaluated on a computer with an Intel Core i5-3317U 1.7GHz). For the mapping task, updating or evaluating the map has computational complexity $O(1)$. Regardless of the size of the map, by exploiting the local property of B-spline, only 16 control points (for $d = 3$) have to be evaluated or updated. For the localisation task the computational cost is similar to the front-end of Cartographer or any other grid-based method that relies on bi-cubic interpolation. The complexity grows linearly with the number of iterations of the nonlinear least squares solver. In the worst case, the computational time is limited by the maximum number of iterations.

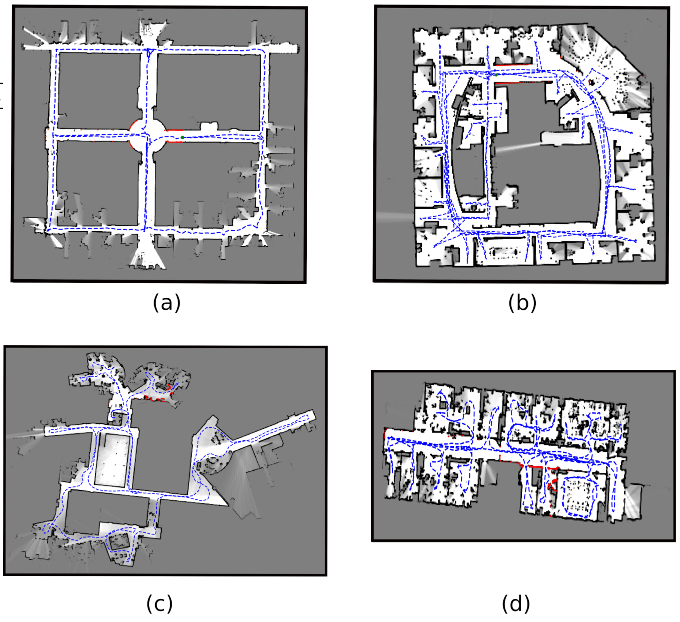

Fig. 5. B-spline SLAM output using the Radish data set [29]: (a) ACES Bulding, (b) Intel Research Lab, (c) MIT CSAIL building, and (d) Freiburg building 69. The blue dashed line is the path travelled by the robot.

VI. Conclusion

This paper presents an online B-spline SLAM strategy for range based sensors. In contrast to classical SLAM solutions that use discrete maps to represent the environment, the proposed solution relies on a B-spline surface map. In general, B-spline maps are more accurate than discrete maps as they do not require discretising floating point measurements into cell resolution. This has a direct impact on the outcome of SLAM. Although we do not perform explicit loop closure, the results using public data sets show that our SLAM framework is able to build accurate maps at sensor rate speed. Quantitative results also show that B-spline SLAM outperform multi-hypothesis grid-based SLAM and it is able to compete with state-of-the-art solutions that perform offline optimisation. The performance of B-spline SLAM can be further improved by adding loop closure detection and offline optimisation.

References

[1] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.

[2] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, “A flexible and scalable slam system with full 3d motion estimation,” in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

TABLE II

COMPARISON USING THE METRICS PROPOSED IN [29]. FOR EACH METRIC, BEST RESULTS ARE HIGHLIGHTED IN BOLD. DASHES INDICATE THAT NO INFORMATION WAS PROVIDED BY THE AUTHORS FOR THAT DATA SET.

	B-spline SLAM	RBPF (50 part)	Cartographer	Two-level opt.	TSDF
Aces					
Absolute translational (m)	0.0404 ± 0.0452	0.060 ± 0.049	0.0375 ± 0.0426	$\mathbf{0.0283 \pm 0.0397}$	**-**
Squared translational (m^2)	0.0036 ± 0.0127	0.006 ± 0.011	0.0032 ± 0.0285	$\mathbf{0.0027 \pm 0.0100}$	**-**
Absolute rotational (deg)	$\mathbf{0.340 \pm 0.392}$	1.2 ± 1.3	0.373 ± 0.469	0.351 ± 0.428	-
Squared rotational (deg^2)	$\mathbf{0.270 \pm 1.657}$	3.1 ± 7.0	0.359 ± 3.696	0.294 ± 1.253	-
Intel					
Absolute translational (m)	0.0262 ± 0.0281	0.070 ± 0.083	0.0229 ± 0.0239	$\mathbf{0.0150 \pm 0.0204}$	-
Squared translational (m^2)	0.0014 ± 0.0066	0.011 ± 0.034	0.0011 ± 0.0040	$\mathbf{0.0009 \pm 0.0009}$	-
Absolute rotational (deg)	0.445 ± 0.969	3.0 ± 5.3	0.453 ± 1.335	$\mathbf{0.390 \pm 0.402}$	-
Squared rotational (deg^2)	$\mathbf{1.137 \pm 6.654}$	36.7 ± 187.7	1.986 ± 23.988	1.629 ± 9.736	-
MIT Killian Court					
Absolute translational (m)	1.0379 ± 2.5719	0.122 ± 0.386^1	0.0395 ± 0.0488	0.0367 ± 0.0473	$\mathbf{0.0276 \pm 0.0235}$
Squared translational (m^2)	7.6918 ± 24.8118	0.164 ± 0.814^1	0.0039 ± 0.0144	0.0031 ± 0.0134	$\mathbf{0.0013 \pm 0.0095}$
Absolute rotational (deg)	0.779 ± 1.246	0.8 ± 0.8^1	0.352 ± 0.353	0.294 ± 0.275	$\mathbf{0.2807 \pm 0.2462}$
Squared rotational (deg^2)	2.160 ± 5.652	0.9 ± 1.7^1	0.248 ± 0.610	0.218 ± 0.439	$\mathbf{0.1394 \pm 0.26865}$
MIT CSAIL					
Absolute translational (m)	$\mathbf{0.0268 \pm 0.0223}$	0.049 ± 0.049^1	0.0319 ± 0.0363	-	-
Squared translational (m^2)	$\mathbf{0.0012 \pm 0.0041}$	0.005 ± 0.013^1	0.0023 ± 0.0099	-	-
Absolute rotational (deg)	$\mathbf{0.315 \pm 0.274}$	0.6 ± 1.2^1	0.369 ± 0.365	-	-
Squared rotational (deg^2)	$\mathbf{0.175 \pm 0.306}$	1.9 ± 17.3^1	0.270 ± 0.637	-	-
Freiburg bldg 69					
Absolute translational (m)	0.0410 ± 0.0315	0.061 ± 0.044^1	0.0452 ± 0.0354	0.0421 ± 0.0349	$\mathbf{0.0382 \pm 0.0292}$
Squared translational (m^2)	0.0027 ± 0.0044	0.006 ± 0.020^1	0.0033 ± 0.0055	0.0029 ± 0.0048	$\mathbf{0.0023 \pm 0.0044}$
Absolute rotational (deg)	$\mathbf{0.420 \pm 0.472}$	0.6 ± 0.6^1	0.538 ± 0.718	0.483 ± 0.571	0.4245 ± 0.4610
Squared rotational (deg^2)	0.399 ± 1.310	0.7 ± 2.0^1	0.804 ± 3.627	0.682 ± 1.533	$\mathbf{0.3926 \pm 1.2308}$

^{1}Odometry was improved using a pre-processing scan-matching step (see [29]).

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec 2016.

[5] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 4387–4393.

[6] A. Caccavale and M. Schwager, "Wireframe mapping for resource-constrained robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[7] L. Pedraza, D. Rodriguez-Losada, F. Matía, G. Dissanayake, and J. V. Miró, "Extending the limits of feature-based slam with b-splines," *Trans. Rob.*, vol. 25, no. 2, p. 353–366, Apr. 2009.

[8] R. T. Rodrigues, A. P. Aguiar, and A. Pascoal, "A b-spline mapping framework for long-term autonomous operations," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3204–3209.

[9] R. T. Rodrigues, N. Tsiogkas, A. P. Aguiar, and A. Pascoal, "B-spline surfaces for range-based environment mapping*," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2020.

[10] R. Smith, M. Self, and P. Cheeseman, *Estimating Uncertain Spatial Relationships in Robotics*. New York, NY: Springer New York, 1990, pp. 167–193.

[11] G. Dissanayake, H. Durrant-Whyte, and T. Bailey, "A computationally efficient solution to the simultaneous localisation and map building (slam) problem," in *2000 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2000, pp. 1009–1014 vol.2.

[12] K. P. Murphy, "Bayesian map learning in dynamic environments," in *In Neural Info. Proc. Systems (NIPS)*. MIT Press, 2000, pp. 1015–1021.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2002, pp. 593–598.

[14] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *in Proceedings of the International Joint Conference on Artificial Intelligence*, ser. IJCAI'03, San Francisco, CA, USA, 2003, p. 1151–1156.

[15] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, Mar 1985, pp. 116–121.

[16] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, Oct 1997.

[17] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, p. 239–256, Feb. 1992.

[18] D. Hahnel, D. Schulz, and W. Burgard, "Map building with mobile robots in populated environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2002, pp. 496–501 vol.1.

[19] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, 2003, pp. 206–211 vol.1.

[20] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2432–2437.

[21] J. Zhao, L. Zhao, S. Huang, and Y. Wang, "2d laser slam with general features represented by implicit functions," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4329–4336, 2020.

[22] H. Xiong, Y. Chen, X. Li, and B. Chen, "A two-level optimized graph-based simultaneous localization and mapping algorithm," *Industrial Robot: An International Journal*, 2018.

[23] R. Ren, H. Fu, and M. Wu, "Large-scale outdoor slam based on 2d lidar," *Electronics*, vol. 8, no. 6, p. 613, 2019.

[24] K. Daun, S. Kohlbrecher, J. Sturm, and O. von Stryk, "Large scale 2d laser slam using truncated signed distance functions," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2019, pp. 222–228.

[25] C. de Boor, *A Practical Guide to Splines*. New York, NY: Springer-Verlag, 1978.

[26] T. Lyche and K. Morken, *Spline Methods*. Norway: Unpublished, 2018.

[27] J. T. Barron, "A general and adaptive robust loss function," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4326–4334.

[28] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[29] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, Sep 2009.