2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

B-spline Surfaces for Range-Based Environment Mapping*

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

***Abstract*— In this paper, we propose a mapping technique that builds a continuous representation of the environment from range data. The strategy presented here encodes the probability of points in space to be occupied using 2.5D B-spline surfaces. For a fast update rate, the surface is recursively updated as new measurements arrive. The proposed B-spline map is less susceptible to precision and interpolation errors that are present in occupancy grid-based methods. From simulation and experimental results, we show that this approach leverages the floating point resolution of continuous metric maps and the fast update/access/merging advantages of discrete metric maps. Thus, the proposed method is suitable for online robotic tasks such as localization and path planning, requiring minor modification to existing software that usually operates on metric maps.**

I. Introduction

Advances in the field of robotics have reached a point where autonomous mobile robots can accomplish a variety of missions from indoor service robotics to ocean sampling. In most scenarios, the robot requires a model of the environment for performing different tasks such as localisation, navigation, path planning, or even to reason about its actions. A common method to model the environment is a map, which is constructed by the robot using its onboard sensors.

Over the years, multiple mapping strategies have been proposed. An initiative to standardize the robot map data representation is presented in the IEEE Standard for robot map data representation [1], which classifies 2D maps in two categories: topological and metric. Topological maps encode the connectivity between places or regions in an environment. Metric maps, on the other hand, retain geometric information of the environment, which allows computing a metric distance between two points of the model. This paper focuses on metric maps, which are useful when precision plays a key role, as is the case with localisation [2] and navigation [3] tasks. Metric maps are further categorised into continuous and discrete map representations.

This work was supported by PDMA-NORTE-08-5369-FSE-000061; IMPROVE POCI-01-0145-FEDER-031823; HARMONY POCI-01-0145-FEDER-031411; SYSTEC - FCT Project UIDB/00147/2020; Flanders Make/VLAIO: SBO MULTIROB, LARSyS - FCT Project UIDB/50009/2020; and EU BluRoses project/GA No. 863619.

[1]Rômulo T. Rodrigues and A. Pedro Aguiar are with Faculty of Electrical Engineering, University of Porto, 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

A continuous metric map uses a set of geometric features for representing the environment. These features have a continuous range of values, i.e., floating point resolution. In [4], the authors model the environment using points, linear segments, and curved lines. The computed map is accurate, but it cannot handle the inherent uncertainty that arises from noisy sensor data. A compact, and yet accurate representation using wireframes is shown in [5]. The method performs well while the assumption that the environment can be represented by line segments holds, e.g., straight walls. To cope with map uncertainty, their solution relies on a particle filter that assigns a particle to each wireframe candidate. More recent approaches as [6], [7] have shown that B-splines curves are a well-suited tool for modeling obstacles of different shapes and sizes. The main drawbacks in most geometric approaches are merging the geometric primitives and the cost to evaluate whether a point in the space belongs to the occupied or free space. *Gaussian Processes* (GPs) maps [8], [9] tackle the fore-mentioned limitations at the expense of computational complexity. In [10], the authors presented for the first time an online GP-mapping. This promising method was shown to be able to map regions at rates close to 10 Hz, which is fast enough for many applications. However, modern sensors can produce data at higher rates, making this method not suitable for data-intense applications.

Discrete metric maps represent the world in a discrete fashion. The most popular metric map, known as occupancy grid maps [3], belongs to this category. It discretises the environment into cells with a specific resolution. Each cell represents the occupancy probability of the represented area. The main advantages are the computational efficiency to update the map, to evaluate the occupancy of a specific point and merge measurements taken at different sensor readings. Most likely, these are the reasons that led the simultaneous localization and mapping (SLAM) community to favor occupancy-maps, e.g., [2], [11], [12]. However, the precision of an occupancy map is limited. Also, its discrete nature does not permit the direct computation of interpolated values or derivatives. Such properties are desirable, as they allow to query the occupancy at a sub-grid cell accuracy. Therefore, a continuous model is often obtained by applying an extra processing step such as bilinear filtering [11] or bicubic interpolation [12]. However, as demonstrated later in this paper, information is lost in the process, leading to errors in terms of accuracy and precision.

The mapping strategy presented in this work represents the environment using continuous B-spline surfaces. The control points of the surface are updated recursively as new range-based measurements arrive. This allows updating the map at fast rates, and, therefore, the method is suitable for online applications. We show that B-spline surface maps have the floating point resolution of continuous metric maps and the low computational complexity of discrete metric maps. This work is motivated by the fact that a more accurate map at the bottom level of a SLAM architecture will improve the overall performance of localization and mapping and other tasks that rely on them such as motion planning and control.

To evaluate the proposed approach, simulated and real-world experiments are conducted. In both cases, the robot is provided with the same positioning information, and a map of the environment is constructed by manually driving the robot around. Common positioning is required so that all the mapping methods can construct the map using the same poses and sensor measurements. This contributes to a fair comparison between the different strategies. The simulations allow for the evaluation of the accuracy of the constructed map against a ground truth representation of the robot environment. The real-world experiments provide qualitative comparison with other well-established literature methods.

The paper is organised as follows. Section II introduces B-splines, the main tool employed in the proposed solution. In section III, the theoretical development of the proposed mapping strategy is presented in detail. Then, in section IV simulation and real-world experiments are presented and their corresponding results are discussed. Finally, section V, concludes the paper, providing final remarks and potential paths for future research on the topic.

II. 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 interchangeable.

B. B-splines

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 is supported by a knot vector $\{t_i\}_{i=0}^{m+d}$, with $t_i \leq t_{i+1}, \forall i$. Given $\mathbf{b}(\tau) = [b_0^d(\tau), \ldots, b_{m-1}^d(\tau)]^T$, it follows from the De Boor's recursive algorithm [13] that

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

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 this definition, we conclude that the following three 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 $b_{\mu-d}^d, \ldots, b_\mu^d$.

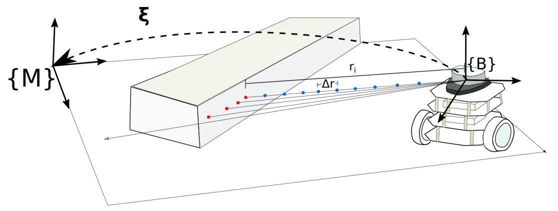

Fig. 1. Problem setup: a robot equipped with a range sensor detects occupied space (red dots) at discrete intervals. The points between the robot and the obstacle are assumed as free space (blue dots) and obtained at sampling intervals of Δr along a beam.

Property 2: (*Local knot*) The B-spline coefficient $b_\mu^d(\tau)$ depends only on the knots $\{t_i\}_{i=\mu}^{\mu+d+1}$.

Property 3: (*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.

A spline surface $\mathbf{s}(\boldsymbol{\tau}) : \mathbb{R}^2 \to \mathbb{R}$, where $\boldsymbol{\tau} = [\tau_x, \tau_y]^T \in \mathbb{R}^2$, is defined by the tensor product of two spline spaces, that is,

$$\mathbf{s}(\boldsymbol{\tau}) = \sum_{i=0}^{m_x-1} \sum_{j=0}^{m_y-1} \mathbf{c}_{i,j} b_i^{d_x}(\tau_x) b_j^{d_y}(\tau_y) \quad (3)$$

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. The tensor product $\mathbf{s}(\boldsymbol{\tau})$ has degree $d = d_x d_y$. Let $C \in \mathbb{R}^{m_x \times m_y}$ be a real matrix with entries $\mathbf{c}_{i,j}$. Then a B-spline surface can be written in matrix form as

$$\mathbf{s}(\boldsymbol{\tau}) = \mathbf{b}_x(\tau_x)^T C \mathbf{b}_y(\tau_y) \quad (4)$$

Now, let *vec* represent the vectorization operator - a linear transformation that stacks the columns of a matrix on top of one another, yielding a single column-vector. Then, the previous tensor product can be written as

$$\begin{aligned} s(\boldsymbol{\tau}) &= \operatorname{vec}\left(\mathbf{b}_x(\tau_x)^T C \mathbf{b}_y(\tau_y)\right) & (5) \\ &= \operatorname{vec}\left(\mathbf{b}_y(\tau_y)^T \otimes \mathbf{b}_x(\tau_x)^T\right) \operatorname{vec}(C) & (6) \\ &= \boldsymbol{\phi}(\boldsymbol{\tau})^T \mathbf{c}, & (7) \end{aligned}$$

where $\boldsymbol{\phi}(\boldsymbol{\tau})^T = \operatorname{vec}(\mathbf{b}_y(\tau_y)^T \otimes \mathbf{b}_x(\tau_x)^T)$, $\mathbf{c} = \operatorname{vec}(C)$, and $\otimes$ stands for the Kronecker product. A B-splinesurface inherits the convex hull property of B-spline curves [14]:

Property 4 (Convex hull): A B-spline surface $\mathbf{s}(\boldsymbol{\tau})$ lies within the convex hull of its control points $(\mathbf{c})$.

III. Range-based mapping using B-splines

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. 1. For the sake of simplicity, assume that the sensor lies at the center of mass of the vehicle.[1] The pose of the robot describes its position and orientation and it is denoted as $\boldsymbol{\xi} = [x, y, \psi]^T$. The

[1]In real applications, the user must provide a rigid transformation from the sensor to the body fixed coordinate frame.

position represents the translation of the origin of $\{B\}$ with respect to $\{M\}$ described in the inertial coordinate frame. The orientation represents the angular misalignment of the x-axis of $\{B\}$ with respect to the x-axis of $\{M\}$.

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 space detected as occupied:

$$^{B}\boldsymbol{\tau}_i^{occ} = r_i \begin{bmatrix} \cos\alpha_i \\ \sin\gamma_i \end{bmatrix}, \quad \forall i = 0, \dots, l-1. \tag{8}$$

Equivalently, let $^{B}\boldsymbol{\tau}_{i,j}^{free}$ be the discrete samples detected as free space, i.e., no obstacle. This corresponds to points along a range beam between the robot and the obstacle, given by

$$^{B}\boldsymbol{\tau}_{i,j}^{free} = n\Delta r \begin{bmatrix} \cos\alpha_i \\ \sin\gamma_i \end{bmatrix}, \forall i = 0, \dots, l-1, \quad j = 0, \dots, r_i/\Delta r - 1, \tag{9}$$

where Δr is an appropriate sampling interval. Figure 1 illustrates both occupied and free space measurements.

A vector $^{B}\boldsymbol{\tau}$ described in the body frame is transformed to the map frame as follows:

$$\boldsymbol{\tau} = R(\psi)^{B}\boldsymbol{\tau} + \begin{bmatrix} x \\ y \end{bmatrix}, \text{ with } R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{10}$$

where $R(\psi)$ is known as rotation matrix.

The remainder of this section presents the theoretical development of the spline mapping strategy proposed here.

A. Mapping

Let M_i and Z_i be discrete random variables that represents 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 ($Z_i = 1$).

The probabilistic map is continuously updated upon reception of the sensor data, which provides the likelihood $p(z_i|m_i)$. Applying Bayes's rule, we have

$$p(M_i = 1|z_i) = \frac{p(z_i|M_i = 1)p(M_i = 1)}{p(z_i)}, \tag{11}$$

$$p(M_i = 0|z_i) = \frac{p(z_i|M_i = 0)p(M_i = 0)}{p(z_i)}. \tag{12}$$

Dividing the first equation by the second yields:

$$\frac{p(M_i = 1|z_i)}{p(M_i = 0|z_i)} = \frac{p(z_i|M_i = 1)}{p(z_i|M_i = 0)} \frac{p(M_i = 1)}{p(M_i = 0)}. \tag{13}$$

The notation is simplified as

$$\operatorname{odd}(m_i|z_i) = \operatorname{odd}(z_i|m_i) \operatorname{odd}(m_i), \tag{14}$$

where the odd of a random variable X, that has exactly two possible outcomes ($X = 0$ or $X = 1$), is defined as

$$\operatorname{odd}(x) = \frac{p(X = 1)}{p(X = 0)} \tag{15}$$

Now, taking the logarithm of both sides leads to the update equation for an occupancy-grid map:

$$\log\operatorname{odd}(m_i)^+ = \log\operatorname{odd}(m_i)^- + \kappa, \tag{16}$$

where we short the notation by substituting $\kappa \equiv \log\operatorname{odd}(z_i|m_i)$ (measurement), $\log\operatorname{odd}(m_i)^- \equiv \log\operatorname{odd}(m_i)$ (prior), and $\log\operatorname{odd}(m_i)^+ \equiv \log\operatorname{odd}(m_i|z_i)$ (posterior). The knowledge of the map is given by the posterior probability, which is obtained by incorporating the measurement to the prior. In occupancy-grid maps, the world is represented by discrete cells. Up to a saturation, the map update in occupancy-grid is given by (16), which iterates the i-th cell according to its corresponding measurement. Here, we propose representing the world by a continuous B-spline surface, that is,

$$\log\operatorname{odd}(m_i) \approx s(\boldsymbol{\tau}_i) = \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}, \tag{17}$$

where $\mathbf{c}$ are the control points that define the spline surface and must be updated upon the reception of new measurements. To obtain the spline map update equation, first, rewrite (16) as

$$s(\boldsymbol{\tau}_i)^+ = s(\boldsymbol{\tau}_i)^- + \kappa \tag{18}$$

$$\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^+ = \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^- + \kappa \tag{19}$$

where, similarly to the original equation, $s(\boldsymbol{\tau}_i)^+$ and $s(\boldsymbol{\tau}_i)^-$ are the posterior and prior log-odd probabilities of the space being occupied. We want to find the control points of the posterior spline map distribution, i.e., $\mathbf{c}^+$. Consider the following error function

$$e_i(\mathbf{c}^+) = \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^+ - \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^- - \kappa, \tag{20}$$

and the cost function

$$J_i(\mathbf{c}^+) = \frac{1}{2}\|e_i(\mathbf{c}^+)\|^2. \tag{21}$$

Given the map prior and a measurement, the map update in (19) is equivalent to finding the control points $\mathbf{c}^+$ that minimize the cost function $J(\cdot)$. In order to do this, we update the control points using the gradient descent method as follows:

$$\mathbf{c}^+ = \mathbf{c}^- - \mu\nabla_{\mathbf{c}^+} J_i(\mathbf{c}^+), \tag{22}$$

where $\mu \in \mathbb{R}^+$. The gradient of the cost function with respect to the control points is

$$\nabla_{\mathbf{c}^+} J_i(\mathbf{c}^+) = \left[\frac{\partial J_i}{\partial \mathbf{c}^+}\right]^T \tag{23}$$

$$= \frac{dJ_i}{de_i}\left[\frac{\partial e_i}{\partial \mathbf{c}^+}\right]^T = e_i(\mathbf{c}^+)\boldsymbol{\phi}(\boldsymbol{\tau}_i) \tag{24}$$

Substituting the gradient back into the recursive equation (22) and the error as defined in (20) yields:

$$\mathbf{c}^+ = \mathbf{c}^- - \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)e(\boldsymbol{\tau}_i) \tag{25}$$

$$= \mathbf{c}^- - \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)[\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^+ - \boldsymbol{\phi}(\boldsymbol{\tau}_i)^T \mathbf{c}^- - \kappa]. \tag{26}$$

With some algebraic manipulation, we obtain

$$[I + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T]\mathbf{c}^+ = [I + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T]\mathbf{c}^- + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\kappa, \tag{27}$$

where I is an identity matrix with appropriate dimensions. The matrix $[I + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T]$ is full rank and, therefore, invertible. Multiplying both sides of the equality by its inverse yields:

$$\mathbf{c}^+ = \mathbf{c}^- + [I + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T]^{-1}\boldsymbol{\phi}(\boldsymbol{\tau}_i)\mu\kappa. \tag{28}$$

Using the Sherman-Morrison formula [15], the following equality holds:

$$[I + \mu\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T]^{-1} = I - \mu\frac{\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}. \tag{29}$$

Replacing the inverse matrix in (28):

$$\begin{aligned}\mathbf{c}^+ &= \mathbf{c}^- + [I - \mu\frac{\boldsymbol{\phi}(\boldsymbol{\tau}_i)\boldsymbol{\phi}(\boldsymbol{\tau}_i)^T}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}]\boldsymbol{\phi}(\boldsymbol{\tau}_i)\mu\kappa && (30)\\ &= \mathbf{c}^- + \boldsymbol{\phi}(\boldsymbol{\tau}_i)[1 - \frac{\mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}]\mu\kappa && (31)\\ &= \mathbf{c}^- + \boldsymbol{\phi}(\boldsymbol{\tau}_i)\frac{\mu}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa && (32)\end{aligned}$$

Applying this result in (20), the mapping error becomes

$$e_i(\mathbf{c}^+) = \boldsymbol{\phi}^T\boldsymbol{\phi}(\boldsymbol{\tau}_i)\frac{\mu}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa - \kappa. \tag{33}$$

In particular, as $\mu \to \infty$, we have that

$$e_i(\mathbf{c}^+) = \lim_{\mu\to\infty}\frac{\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2\mu}{1 + \mu\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa - \kappa = 0 \tag{34}$$

We conclude that as the gain μ, which can be made arbitrarily large, goes to infinity the error converges to zero. Finally, applying this result in (32), the map update equation upon reception of a measurement at $\boldsymbol{\tau}_i$ is given by

$$\mathbf{c}^+ = \mathbf{c}^- + \frac{\boldsymbol{\phi}(\boldsymbol{\tau}_i)}{\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa. \tag{35}$$

B. Implementation notes

As presented in Sec. II, the tensor product $\boldsymbol{\phi}(\boldsymbol{\tau})$ is the Kronecker product of two B-splines of degree d_x and d_y. Based on the Property 1 (Local Support) of B-splines, each of these functions has at most $d_x + 1$ and $d_y + 1$ non-null B-splines coefficients. Therefore, $\boldsymbol{\phi}(\boldsymbol{\tau})$ has at most $(d_x + 1)(d_y + 1)$ non-null B-splines coefficients. In our implementation, we use cubic splines, i.e., $d_x = d_y = 3$. This allows some degree of freedom for the spline to cope with high curvatures. At the same time, at most 16 control points are updated by a measurement, which keeps the computational cost bounded within an acceptable time. The knots are evenly spaced on a grid. The space between the knots, called knot interval, is associated with the map resolution. The knot interval and the size of the map defines the memory required. We do not store the knot vector, but only the control points associated to them in a grid matrix. Therefore, smaller knot intervals require more control points for a constant map size. Since the knots are evenly spaced, Property 3 (Continuity) holds. This is an important feature, as many map applications require a continuous and smooth map. The recursive equation (1) was expanded for a constant knot interval Δt. Then, the Kronecker product of two splines computed, obtaining a closed form non-recursive manner to evaluate $\boldsymbol{\phi}(\boldsymbol{\tau})$.

Similarly to occupancy-grid maps, we saturate the belief of the map. For this purpose, we apply Property 4 (Convex hull) as follows

$$\mathbf{c}^+ = \min(\max(\mathbf{c}^- + \frac{\boldsymbol{\phi}(\boldsymbol{\tau}_i)}{\|\boldsymbol{\phi}(\boldsymbol{\tau}_i)\|^2}\kappa, c_{min}), c_{max}). \tag{36}$$

This ensures that the surface is bounded within the interval $[c_{min}, c_{max}]$. For the results shown in this paper, we use the following parameters values: $c_{max} = 100$, $c_{min} = -100$, $\Delta r = 1.41\Delta t$, where Δt is the knot interval. Also, we set $\kappa = 0.9$ for measurements corresponding to occupied space, and $\kappa = 0.3$ for measurements corresponding to free space. An algorithmic representation of the whole map update process can be seen in Algorithm 1.

Algorithm 1 B-spline map update algorithm.

```
Input: Pose $\boldsymbol{\xi}$, Range Scans $(r_i)_{i=0}^{l-1}$
1: Remove wrong measurements
2: Transform range to coordinates: (8)
3: Detect free space: (9)
4: Transform free and occupied space to global coordinate frame: (10)
5: Update the B-spline map: (36)
```

IV. Experiments

In this section, the proposed mapping strategy is compared against other methods described in the literature. The comparison is performed using both simulated and real-world data of a LiDAR sensor mounted on a mobile robot. The simulations are used to discuss quantitative performance. While using a real-robot, we show qualitative results. All the experiments were run using a notebook Intel Core i5-3317U CPU @ 1.70 GHz, 8 GB RAM.

For a fair comparison, we implemented our proposed solution and a vanilla version of the occupancy-grid mapping algorithm in Python. Both are available in our repository[2]. The occupancy-grid implementation follows the Algorithm 1, but we use (16) to update the discrete cells m_i and ray rasterization (Bresenham's line algorithm) for detecting the free space.

A. Simulations

The proposed method is compared against occupancy-grid based strategies in a controlled environment, using extensive computer simulations. To this end, synthetic range data is generated that correspond to a sensor having 360 beams equally spaced using one degree intervals. The maps are constructed using the artificial data. The occupancy-grid

[2]https://github.com/C2SR/spline_slam/

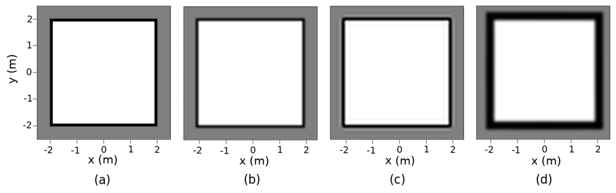

Fig. 2. Mapping result for an artificially generated square room using: (a) occupancy-grid, (b) occupancy-grid + bilinear interpolation, (c) occupancy-grid + bicubic interpolation, and (d) proposed B-spline map. The robot is in the center of the room.

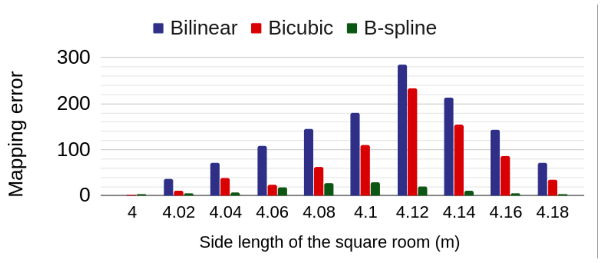

Fig. 3. Mapping error using the room scenario shown in Fig. 2. Both the cell resolution and knot vector grid have a resolution of 0.1 m. Throughout the scenarios, the shortest distance from the robot to the wall varies from 2 m to 2.09 m.

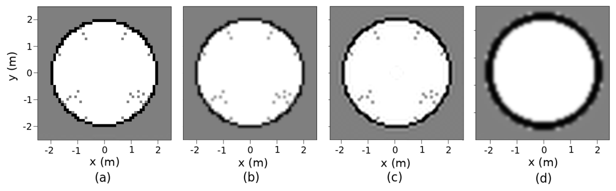

Fig. 4. Mapping result for an artificially generated circular room using: (a) occupancy-grid, (b) occupancy-grid + bilinear interpolation, (c) occupancy-grid + bicubic interpolation, and (d) proposed B-spline map.

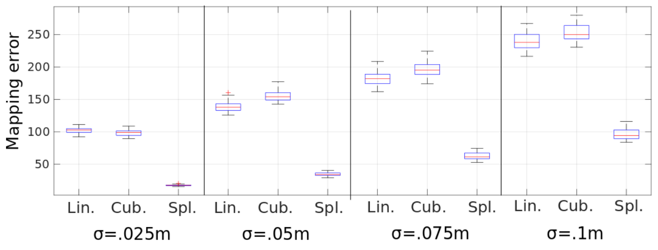

Fig. 5. Statistical analysis using noisy sensor data for the scenario shown in Fig. 4. The noise follows a zero-mean Gaussian distribution. Different standard deviations were considered. *Lin.*, *Cub.*, and *Spl.* stands for the bilinear, the bicubic, and the B-spline map, respectively.

map cell resolution and the B-spline map knot interval are both set to 0.1 m - therefore, having the same memory storage requirements. Then, for obtaining a continuous map, bilinear and bicubic interpolations are applied to the resulting occupancy-grid map, as proposed in [11] and [12], respectively. The maps are finally normalized into the floating interval -1 (free) to 1 (occupied).

In order to compare the mapping accuracy of the continuous maps, the following metric is used:

$$\sum_{i=0}^{n-1}[1 - M_{cont}(\boldsymbol{\tau}_i)]^2, \tag{37}$$

where M_{cont} is a continuous map. The metric, which is called here the mapping error, translates as the alignment between the measurements detected as obstacles and the actual continuous map belief. It is often used in localisation for estimating the relative displacement between consecutive readings.

The first simulated environment evaluates the limitations of a continuous map obtained from an occupancy-grid map versus the inherent continuous B-spline map. The robot stands still in the center of a square room, while taking measurements. There were enough sensor readings to stabilize the map cells/control points. The resulting maps for a square room of side length 4 m can be seen in Fig. 2. The methods managed to accurately map the given environment. Then, to analyze the discretisation error that rises in occupancy grid mapping, we varied the side length of the square room from 4 m to 4.18 m. Figure 3 shows the mapping error. It can be see that the mapping error using the proposed approach is consistently lower than any of the other approaches. When the length of the square is 4 m the walls are aligned with the cell representation of the occupancy grid. Therefore, the occupancy grid based methods are able to achieve null error, which is slightly better than the spline map. Then, as the length increases, there is an increasing offset between the walls and the cells that represent them in the discrete map. The critical point occurs around half-cell resolution. The interpolation methods fail to capture the reality. In essence, the position within a cell which corresponds to a measurement is not registered by occupancy-grid map approaches. Meanwhile, the error in the B-spline map is considerable lower because when building the map, it is considering the exact place where the obstacle was detected.

The second simulated environment assesses the impact of noise in the range measurements provided by the sensor. This time, the robot stands still at the center of a circular space of 2 meters radius. The sensor range data is perturbed with zero mean Gaussian noise. The standard distribution of the noise varies from .025 m to .1 m, which corresponds from a quarter to one cell/knot interval length. The final map was built after 500 sensor readings. The resulting maps for the noiseless case are shown in Fig. 4. The occupancy based methods have regions of unknown space caused by their discrete nature and due to the ray-casting used, where sensor rays are not intersecting all the cells. The B-spline map does not present the gaps because a measurement has impact on nearby regions. The statistical analysis for the noisy case can be seen in Fig. 5. In total, 33 simulations for each set of map and standard deviation were performed. The proposed method behaves better in all the scenarios. A reason for that is because B-splines naturally filters the noise while building the map.

B. Real-data

The results using real-data were obtained using a Turtlebot3 equipped with a LiDAR sensor. The robot was teleoper-

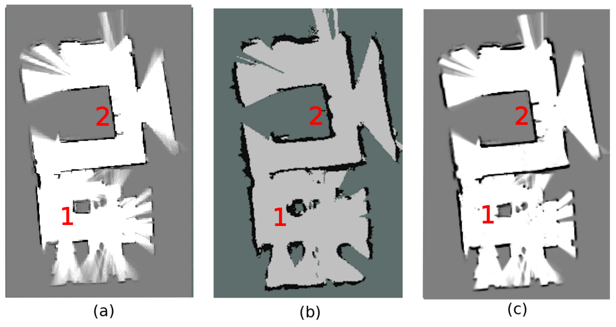

Fig. 6. Experimental results using turtlebot/LiDAR setup in the corridors of FEUP. The results shown correspond to the output of occupancy-grid (a), Fast-GPOM (b), and the proposed B-spline map (c). The proposed approach is representing the environment in an accurate manner being able to capture even small obstacles. Annotations are depicted in red numbers.

TABLE I

RUNNING-TIME FOR BUILDING THE MAPS SHOWN IN FIG 6

	Occupancy-grid	Fast-GPOM	Spline-Map
Time (ms)	16.34	142.60	11.48

ated through the corridors of the Faculty of Engineering of the University of Porto (FEUP). For generating the maps, first we recorded a bagfile. Then, we estimated the pose of the robot using GMapping [2], [16]. The estimated pose and LiDAR measurements were fed into different mapping techniques - all of them implemented in Python. The maps were generated using 5 cm resolution for both the discrete cells and knot spacing. Figure 6(a) shows the occupancy-grid map obtained using our vanilla implementation. Figure 6(b) illustrates the output of Fast-GPOM [10], a continuous Gaussian Process map. Figure 6(c) shows the map obtained using the proposed B-spline surface method. A few annotations are shown in red numbers. The three methods perform rather well. A careful inspection shows that the upper right section of the Fast-GPOM map is slight deformed, representing the region larger than it actually is. As highlighted in annotation 1, Fast-GPOM is not able to register well features with high curvature. Although this is a typical limitation of continuous maps, our method is able to capture sharp regions significantly better. The proposed B-spline map is also capable of handling small objects. For example, annotation 2 corresponds to the three legs of a large bench, each leg is 5 cm wide.

The average time per iteration for building the maps is shown in Table I. The results show that the B-spline map is the fastest one, followed by the occupancy-grid map. This was not expected, because an occupancy-grid map updates one cell per measurement, while the proposed strategy updates 16 control points per measurement. It is most likely that our B-spline map implementation is more optimized than our occupancy-grid map implementation. While both aforementioned methods have computational complexity $O(1)$ per measurement update, Gaussian Process maps have computational complexity $O(n^3)$ [17], where n is the size of the sub-matrix of the kernel to be updated. This explains the high average time per iteration.

V. CONCLUSION

This work presented a novel 2.5D continuous mapping approach based on B-spline surfaces. The proposed method allows for more accurate representations of the world leveraging floating point resolution of continuous metric maps with the fast update/access/merging advantages of discrete metric maps. The B-spline map was evaluated in simulations and on real world data. In terms of error in representation, it performed better than a vanilla implementation of the occupancy grid map. In terms of speed, it performed better than a state of the art implementation of a Gaussian Process mapping technique that also represents the world in a continuous fashion. Potential future directions of this work include the extension to a full SLAM, using the continuous properties of B-spline to estimate the localisation of the robot.

REFERENCES

[1] "IEEE standard for robot map data representation for navigation," *1873-2015 IEEE Standard for Robot Map Data Representation for Navigation*, pp. 1–54, Oct 2015.

[2] 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.

[3] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989.

[4] R. Vazquez-Martin, P. Nunez, A. Bandera, and F. Sandoval, "Curvature-based environment description for robot navigation using laser range sensors," *Sensors*, vol. 9, no. 8, pp. 5894–5918, 2009.

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

[6] L. Pedraza, G. Dissanayake, J. V. Miro, D. Rodriguez-Losada, and F. Matia, "Bs-slam: Shaping the world." in *Robotics: Science and Systems*, 2007, pp. 1–8.

[7] 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)*. IEEE, 2018, pp. 3204–3209.

[8] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[9] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, no. 2, pp. 273–290, 2018.

[10] Y. Yuan, H. Kuang, and S. Schwertfeger, "Fast gaussian process occupancy maps," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1502–1507.

[11] 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.

[12] 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.

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

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

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. USA: Cambridge University Press, 2007.

[16] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, winter 2010.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.