Improved Non-linear Spline Fitting for Teaching Trajectories to Mobile Robots

Christoph Sprunk Boris Lau Wolfram Burgard

***Abstract*— In this paper, we present improved spline fitting techniques with the application of trajectory teaching for mobile robots. Given a recorded reference trajectory, we apply non-linear least-squares optimization to accurately approximate the trajectory using a parametric spline. The fitting process is carried out without fixed correspondences between data points and points along the spline, which improves the fit especially in sharp curves. By using a specific path model, our approach requires substantially fewer free parameters than standard approaches to achieve similar residual errors. Thus, the generated paths are ideal for subsequent optimization to reduce the time of travel or for the combination with autonomous planning to evade obstacles blocking the path. Our experiments on real-world data demonstrate the advantages of our method in comparison with standard approaches.**

I. Introduction

In the recent years, mobile transportation platforms became more and more popular in industrial applications. The majority of them are so-called automated guided vehicles (AGVs) designed to carry loads on predefined paths, often marked by magnetic or optical strips. Using path planners for autonomous motion, however, is usually more flexible because vehicles can easily be assigned to new goals and directly cope with unexpected obstacles. The commercial KIVA system, for example, uses A* planning on a grid to control autonomous vehicles in warehouses and distribution centers [1]. However, compared to AGVs, the movement of such autonomously navigating systems is less predictable, which is sometimes not desired in production sites shared with human workers.

Automation of flexible production processes with small lot sizes often requires systems that can easily be assigned to new paths — even by non-experts and without changing the environment. A natural approach is to record reference trajectories and to fit continuous paths to them. If desired, the paths can be further optimized to reduce travel time or to allow the robot to autonomously deviate from the path in exceptional circumstances, such as unexpected obstacles blocking it. To facilitate efficient optimization and to achieve robustness to noise, a key challenge is to achieve accurate path fits with a small number of parameters in the model.

This paper presents a novel approach to improved trajectory teaching for mobile robots by non-linear fitting of a specific path model (see Fig. 1). It substantially reduces the number of parameters required to achieve the same fitting accuracy as standard approaches. We describe our model and the method to efficiently fit it to reference paths, to optimize the number and location of control points, and to refine spline fits with non-linear optimization.

All authors are with the computer science department at the University of Freiburg, Germany, {sprunkc, lau, burgard}@informatik.uni-freiburg.de.

This work has partly been supported by the European Commission under FP7-248258-First-MM, FP7-248873-RADHAR, and FP7-260026-TAPAS.

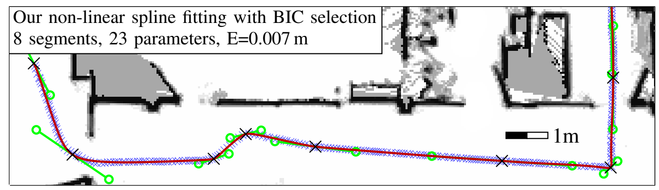

Fig. 1. Fitting our path model (red) to a reference trajectory (blue). Our non-linear optimization places control points in curve apices and adjusts their position (crosses) and tangents (circles).

After discussing related work in Sect. II, we formalize basic spline fitting in Sect. III. While Sects. IV–VI present our novel methods for non-linear spline fitting, control point optimization, and path refinement, Sects. VII and VIII present experimental results and discuss application scenarios.

II. Related Work

Trajectory teaching has received considerable attention in areas like programming by demonstration, manipulation, and humanoid robots. Calinon *et al.* for example combine Hidden Markov Models with Gaussian mixture regression to generalize from multiple gesture demonstrations [2]. Several authors have applied spline fitting to generate smooth trajectories from discrete reference data points, e.g., for manipulation imitation [3], foot step planning [4], to represent handwriting motions [5], or to analyze human trajectories [6].

Basic spline fitting can be performed using linear least-squares minimization given fixed correspondences between the reference data points and the internal parameter of the spline, which drastically limits the expressiveness of the spline. Wang *et al.* presented an error measure to fit B-splines to point cloud data without such correspondences [7]. For our application, we propose a novel error measure especially suited for non-linear spline-fitting with sparse control points.

For baseline comparison we use basic spline fitting, as done by Hwang *et al.* for manually drawn robot paths [8]. This paper presents an approach to non-linear spline fitting of a specific path model. Compared to standard approaches, it requires substantially fewer parameters to achieve the same accuracy. We also present several application scenarios that exploit this property to further optimize the fitted paths. Like the approach by Macfarlane and Croft, our model uses quintic splines to avoid curvature discontinuities [9].

III. Basic Splines and Linear Fitting Techniques

In mobile robotics, odd-ordered Bézier splines are a popular parametric path representation, since they can be used to smoothly connect a set of waypoints as shown in Fig. 2.

A spline segment $\hat{\mathbf{s}}(u)$ is a polynomial curve of order n, defined over an internal parameter $u \in [0,1]$. In the Hermite form, a spline segment is defined by control points $\mathbf{p}_i$ at its start ($\mathbf{p}_s$) and end ($\mathbf{p}_e$). Each $\mathbf{p}_i$ has $K = \frac{n+1}{2}$ parameters $\mathbf{p}_i^k$, with $k = 0, \ldots, K-1$. The $\mathbf{p}_i^k$ are vectors with one component per spline dimension, and specify the k-th derivative of $\hat{\mathbf{s}}(u)$ at the start ($u=0$) and end ($u=1$) of the segment. $\hat{\mathbf{s}}(u)$ is then given by the linear combination

$$\hat{\mathbf{s}}(u) = \sum_{k=0}^{K-1} h_s^k(u) \cdot \mathbf{p}_s^k + h_e^k(u) \cdot \mathbf{p}_e^k \,, \tag{1}$$

where $h_s^k(u)$ and $h_e^k(u)$ are polynomials called Hermite basis functions. They are obtained by solving

$$\hat{\mathbf{s}}^{(k)}(0) = \mathbf{p}_s^k,\ \ \hat{\mathbf{s}}^{(k)}(1) = \mathbf{p}_e^k, \quad k = 0, \ldots, K-1, \tag{2}$$

where $\hat{\mathbf{s}}^{(k)}$ is the k-th derivative of the polynomial $\hat{\mathbf{s}}$. By factoring out the $\mathbf{p}_s^k, \mathbf{p}_e^k$ we obtain the basis functions for cubic, quintic and septic splines shown in the following table.

	Cubic (K=2)	Quintic (K=3)	Parameter
h_s^0	$2u^3 - 3u^2 + 1$	$-6u^5 + 15u^4 - 10u^3 + 1$	$\mathbf{p}_s^0$
h_s^1	$u^3 - 2u^2 + u$	$-3u^5 + 8u^4 - 6u^3 + u$	$\mathbf{p}_s^1$
h_s^2		$-\frac{1}{2}u^5 + \frac{3}{2}u^4 - \frac{3}{2}u^3 + \frac{1}{2}u^2$	$\mathbf{p}_s^2$
h_e^0	$-2u^3 + 3u^2$	$6u^5 - 15u^4 + 10u^3$	$\mathbf{p}_e^0$
h_e^1	$u^3 - u^2$	$-3u^5 + 7u^4 - 4u^3$	$\mathbf{p}_e^1$
h_e^2		$\frac{1}{2}u^5 - u^4 + \frac{1}{2}u^3$	$\mathbf{p}_e^2$

	Septic (K=4)	
h_s^0	$20u^7 - 70u^6 + 84u^5 - 35u^4 + 1$	$\mathbf{p}_s^0$
h_s^1	$10u^7 - 36u^6 + 45u^5 - 20u^4 + u$	$\mathbf{p}_s^1$
h_s^2	$2u^7 - \frac{15}{2}u^6 + 10u^5 - 5u^4 + \frac{1}{2}u^2$	$\mathbf{p}_s^2$
h_s^3	$\frac{1}{6}u^7 - \frac{2}{3}u^6 + u^5 - \frac{2}{3}u^4 + \frac{1}{6}u^3$	$\mathbf{p}_s^3$
h_e^0	$-20u^7 + 70u^6 - 84u^5 + 35u^4$	$\mathbf{p}_e^0$
h_e^1	$10u^7 - 34u^6 + 39u^5 - 15u^4$	$\mathbf{p}_e^1$
h_e^2	$-2u^7 + \frac{13}{2}u^6 - 7u^5 + \frac{5}{2}u^4$	$\mathbf{p}_e^2$
h_e^3	$\frac{1}{6}u^7 - \frac{1}{2}u^6 + \frac{1}{2}u^5 - \frac{1}{6}u^4$	$\mathbf{p}_e^3$

Typically, a spline curve is a concatenation of multiple spline segments. When joining M segments $\hat{\mathbf{s}}_i$, $i \in 0, \ldots, M-1$, the resulting curve $\mathbf{s}(u)$ is defined over $[0, M]$ and given by $\mathbf{s}(u) = \hat{\mathbf{s}}_i(u - i)$. Here, $\hat{\mathbf{s}}_i$ with $i = \lfloor u \rfloor$ is the "active" segment for a certain u, and specified by $\mathbf{p}_s^k = \mathbf{p}_i^k$ and $\mathbf{p}_e^k = \mathbf{p}_{i+1}^k$. Since adjacent segments share control points, the curve and its derivatives are continuous up to the K–1-th derivative.

A. Linear least-squares spline fitting

Given a reference path $\mathbf{z}(t)$, we want to find an accurate parametric approximation with as few parameters as possible. $\mathbf{z}(t)$ is given by the robot position $\mathbf{z}_t = \langle x_t, y_t \rangle$ at each discrete time step $t = 0, \ldots, N-1$. We compute the cumulative length of the piecewise linear path given by $\mathbf{z}(t)$ as $l_t = \sum_{i=1}^{t} \|\mathbf{z}_i - \mathbf{z}_{i-1}\|$. We assume that the $\mathbf{z}_t$ have been pruned to have a minimum distance $l_t - l_{t-1} > \tau_l$ for all t. To approximate $\mathbf{z}(t)$ with a spline, we assign each $\mathbf{z}_t$ a corresponding u_t by linear interpolation with respect to the arc length, $u_t = M \cdot (l_t / l_{N-1})$.

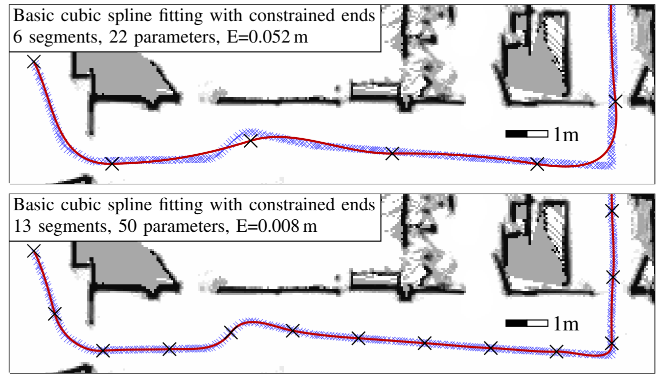

Fig. 2. Linear least-squares fitting of cubic splines with constrained start/end points. With 22 parameters (top) this overly smoothens the corners, causing a high residual error. With 50 parameters (bottom) the error is comparable to our method with 23 parameters, as shown in Fig. 1.

For the basic linear least-squares fit we construct a matrix X with one row $\mathbf{x}_t$ for each data point $\mathbf{z}_t$. The rows contain the Hermite basis functions for the corresponding u_t. For one-dimensional splines these rows are given by

$$\mathbf{x}_t = \Big(\underbrace{0 \ldots 0}_{i \cdot K} \quad h_s^0 \ldots h_s^{K-1} \quad h_e^0 \ldots h_e^{K-1} \quad \underbrace{0 \ldots 0}_{(M-i-1) \cdot K} \Big) \,,$$

where each h^k is a function of $u_t - i$ and $i = \lfloor u \rfloor$ the index of the segment active for u_t. For splines with two or more independent dimensions, $\mathbf{x}_t$ is expanded accordingly. The spline $\mathbf{s}(u)$ at $u = u_t$ is then given by $\mathbf{s}(u_t) = \mathbf{x}_t \cdot \mathbf{p}$, where

$$\mathbf{p} = \big(\mathbf{p}_0^0 \cdots \mathbf{p}_0^{K-1} \quad \ldots \quad \mathbf{p}_M^0 \cdots \mathbf{p}_M^{K-1} \big)^T \tag{3}$$

is the vector of control point parameters. To fit the spline to the reference path $\mathbf{z}$, we can solve the corresponding linear least squares problem in closed form,

$$\tilde{\mathbf{p}} = \arg\min_{\mathbf{p}} \|\mathbf{z} - X\mathbf{p}\| = \left(X^T X \right)^{-1} X^T \mathbf{z} \,. \tag{4}$$

The parameters $\tilde{\mathbf{p}}$ define a spline $\mathbf{s}(u)$ that minimizes the sum of the squared residual errors, $r^2 = \sum_{t=0}^{N-1} \|\mathbf{z}_t - \mathbf{s}(u_t)\|^2$.

B. Constraining start and end of the fitted paths

Applications like mobile manipulation can require high accuracy for the start and end positions. We constrain the spline by removing the corresponding model parameters $\mathbf{p}_0^0$ and $\mathbf{p}_M^0$ from $\mathbf{p}$ and the respective columns from X. Instead, the locations $\mathbf{z}_0$ and $\mathbf{z}_{N-1}$ are put in a constant vector $\mathbf{b}$ for a modified spline fit, $\tilde{\mathbf{p}} = \arg\min_{\mathbf{p}} \|\mathbf{z} - (X\mathbf{p} + \mathbf{b})\|$. The elements of $\mathbf{b}$ depend on the u_t, and are given by

$$\mathbf{b}_t = \delta_{i=0} \cdot h_s^0(u_t - i) \cdot \mathbf{z}_0 + \delta_{i=M-1} \cdot h_e^0(u_t - i) \cdot \mathbf{z}_{N-1} \,,$$

where $\delta_C = 1$ if the condition C in the index is true, and zero otherwise. Furthermore, to constrain the start and end orientations to specified values θ_0 and θ_{N-1}, the first derivative $\mathbf{s}'(u)$ at the start and end has to meet the conditions

$$\mathbf{s}'(0) = e_0 \begin{pmatrix} \cos\theta_0 \\ \sin\theta_0 \end{pmatrix} \,, \quad \mathbf{s}'(M) = e_M \begin{pmatrix} \cos\theta_{N-1} \\ \sin\theta_{N-1} \end{pmatrix} \,, \tag{5}$$

where e_0, e_M are scalar elongation factors that scale the tangent length. Now, $\mathbf{p}_0^1$ and $\mathbf{p}_M^1$ are removed from the

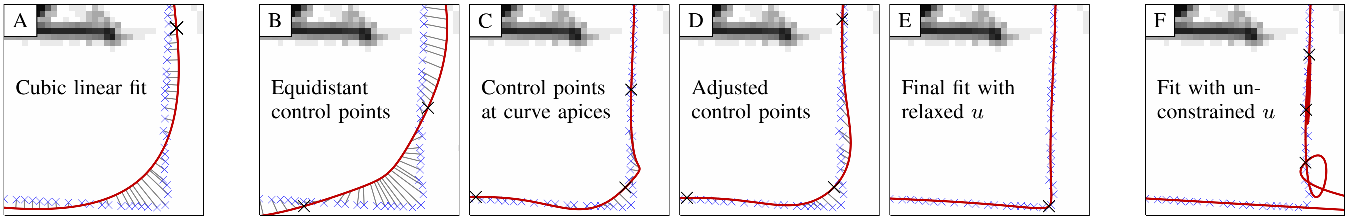

Fig. 3. Spline fits (red) and residual errors (gray lines) for a given reference path (crosses). The plots show the bottom-right corner (1.7×1.8 m) of the map in Fig. 1. (A): linear fit of a basic cubic spline. (B)-(E): different stages of our spline fits. (F): problem when fitting without any constraints on u.

parameter vector $\mathbf{p}$, and replaced by e_0 and e_M. The corresponding coefficients in X are $\delta_{i=0} \cdot h_s^1(u_t - i) \cdot \mathbf{s}'(0)$ and $\delta_{i=M-1} \cdot h_e^1(u_t - i) \cdot \mathbf{s}'(M)$, respectively. Since e_0 and e_M are the same for x and y, the rows and columns of $\mathbf{z}$, $\mathbf{p}$, and X are interleaved for x and y, and the entries for the elongation factors are unified.

Solving the least squares fit for the modified X, $\mathbf{p}$, and $\mathbf{b}$ yields a spline that obeys the constraints mentioned above. Nevertheless, the accuracy of the fit depends on the number of spline segments as shown in Fig. 2. Especially in sharp corners and curves with small radii, the errors can be very high as shown in Fig. 3 (A). In the next sections we propose improvements over the basic spline fitting to reduce the number of parameters and the fitting error at the same time.

IV. Non-linear fits with our path model

This section proposes least-squares fitting for the path model introduced by Lau *et al.* [10]. It is based on quintic splines and reduces the number of parameters with heuristics. For 2D splines it needs 3 instead of 6 parameters per control point, which substantially reduces the computational load for optimization.

Similar to the basic splines, the segments of this model connect a set of consecutive waypoints $\mathbf{p}_i^0$. The first derivative, i.e., the tangent of the spline at the waypoints, is controlled by a heuristic and given by

$$\mathbf{p}_i^1 = e_i \cdot \tfrac{1}{2} \left(\frac{\mathbf{d}_{i-1}}{\|\mathbf{d}_{i-1}\|} + \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|} \right) \cdot \tfrac{1}{2} \min\{\|\mathbf{d}_{i-1}\|, \|\mathbf{d}_i\|\}\,, \quad (6)$$

where $\mathbf{d}_i = \mathbf{p}_{i+1}^0 - \mathbf{p}_i^0$ is the vector between the start and end point of segment i. The e_i are scalar elongation factors that scale the normed tangents at each control point.

The parameters $\mathbf{p}_i^2$ specifying the second derivative are determined by a heuristic that mimics the behavior of cubic splines, but overcomes their curvature discontinuities:

$$\mathbf{p}_i^2 = \frac{\|\mathbf{d}_i\|}{\|\mathbf{d}_{i-1}\| + \|\mathbf{d}_i\|} \lim_{u \nearrow i} \mathbf{s}_h''(u) + \frac{\|\mathbf{d}_{i-1}\|}{\|\mathbf{d}_{i-1}\| + \|\mathbf{d}_i\|} \lim_{u \searrow i} \mathbf{s}_h''(u)\,, \quad (7)$$

where $\mathbf{s}_h''$ is the piecewise linear second derivative of the cubic spline given by $\mathbf{p}_i^0$ and $\mathbf{p}_i^1$. With these heuristics, a quintic spline is fully specified by the waypoints $\mathbf{p}_i^0$ and the elongation factors e_i. Thus, it has 3 parameters per control point in the 2D case, whereas a generic cubic spline has 4, and a quintic spline has 6 parameters per control point. Adding constraints for the start and end pose is done in the same way as for the basic splines.

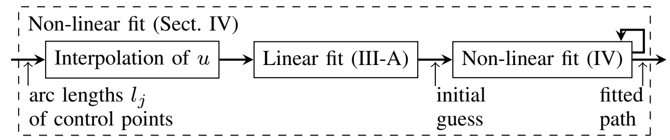

Fig. 4. Non-linear fit of our path model. After interpolating u, we perform a linear least-squares fit of a basic spline. The resulting control points are used as initial guess for the non-linear optimization of our path model.

This quintic path model has shown to be effective in the context of path optimization in various environments and applications. For more details please refer to [10], [11].

A. Fitting with non-linear optimization

To compute splines with our path model, we define a conversion function f, that transforms the parameter vector $\mathbf{p}^+ = \left(\mathbf{p}_1^0 \ldots \mathbf{p}_{M-1}^0 \quad e_0 \ldots e_M\right)^T$ to a basic quintic parameter vector according to Eq. (6) and (7). The least-squares fit with start and end constraints as before is then given by

$$\tilde{\mathbf{p}}^+ = \arg\min_{\mathbf{p}^+} \left\| \mathbf{z} - X \cdot f(\mathbf{p}^+) + \mathbf{b} \right\| . \quad (8)$$

Since f is non-linear, the problem cannot be solved in closed form anymore. Instead, we employ optimization using the Levenberg-Marquardt algorithm. A good initial guess is obtained by computing a linear fit as described in Sect. III-A to initialize the $\mathbf{p}_i^0$ and e_i. The e_i are determined from the $\mathbf{p}_i^1$ by solving Eq. (6) accordingly. For an overview, see also Fig. 4. An exemplary output is shown in Fig. 3 (B).

V. Choosing Control Points

The spline fits in Sections III-A and IV-A optimize the parameters of the control points $\mathbf{p}_i$, but their position along the spline has been determined by the linear interpolation of u for the whole path. This section proposes a method to place the control points of our path model in curve apices, which substantially improves the fit quality.

A. Estimating the location of curve apices

We seek to automatically find curve apices in our training data and denote their increasing cumulative arc length by $l_j, j = 1, \ldots, J$. To detect these points, we fit a basic spline $\mathbf{s}_c(u)$ to the data, and compute its curvature function $c(u)$, which is the reciprocal value of the curve radius at every point on the spline. The curve apices correspond to extremal values of $c(u)$, which are identified by sign changes in the derivative $c'(u)$ where $|c(u)| > \tau_c$, as shown in Fig. 5. The curvature and its derivative are given by

$$c(u) = \left(\frac{\mathbf{s}_c' \times \mathbf{s}_c''}{\|\mathbf{s}_c'\|^3} \right), \; c'(u) = \frac{\mathbf{s}_c' \times \mathbf{s}_c'''}{\|\mathbf{s}_c'\|^3} - \frac{3\,(\mathbf{s}_c' \times \mathbf{s}_c'')\,(\mathbf{s}_c' \cdot \mathbf{s}_c'')}{\|\mathbf{s}_c'\|^5}\,,$$

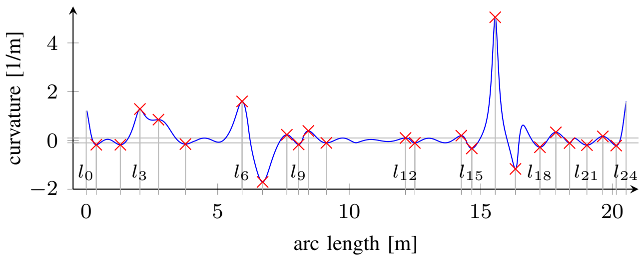

Fig. 5. Curvature of a septic spline $\mathbf{s}_c(u)$ that was linearly fitted to the data points shown in Fig. 1. The crosses mark the detected extrema l_j after thresholding and approximate the position of curve apices along the spline.

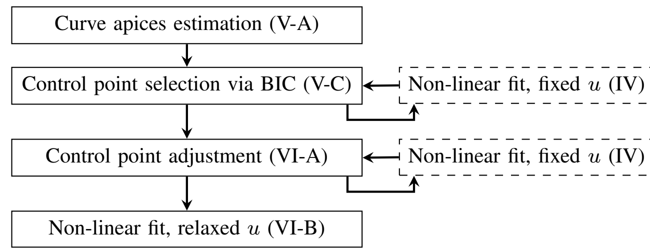

Fig. 6. Overview of our approach. The individual steps are described in the indicated sections. For the non-linear fit (dashed) see also Fig. 4.

where $\mathbf{a} \times \mathbf{b} = a_x b_y - b_x a_y$ for the x and y components of the 2D spline $\mathbf{s}_c(u)$ and its derivatives. Again, we dropped the dependency of $\mathbf{s}_c(u)$ on u for readability.

Since $c'(u)$ depends on the third derivative $\mathbf{s}_c'''(u)$, we use a septic spline for $\mathbf{s}_c(u)$, for which $\mathbf{s}_c'''(u)$ is continuous.

We associate the location of each curve apex j with the arc length l_j of the closest point on the piece-wise linear interpolation of the reference data $\mathbf{z}_t$. The start and end point are treated like curve apices with $l_0 = 0$ and $l_{J+1} = l_{N-1}$, respectively (see Fig. 5). We can "anchor" control points of our spline to these l_j by associating them with integer u values. Then, the spline parameter u_t for a data point $\mathbf{z}_t$ at arc length l_t between two anchored control points with indices j, $j+1$ and arc lengths l_j, l_{j+1} is given by the interpolation

$$u_t = j + \frac{l_t - l_j}{l_{j+1} - l_j}, \text{with } l_j \leq l_t < l_{j+1}\,. \tag{9}$$

B. Bayesian Information Criterion for control point selection

Creating control points for all detected curve apices can yield an overly complex spline model. To find a good trade-off between the number of control points and accuracy, we propose an error-driven model selection procedure. It is based on the Bayesian Information Criterion (BIC),

$$\mathrm{BIC} = -2 \log L + \hat{K} \log N, \tag{10}$$

where L is the data likelihood given the model, $\hat{K}$ is the number of free parameters in the model, and N the number of data samples. The likelihood of a spline fit is a function of the fitting error $r^2 = \sum_{t=0}^{N-1} \|\mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t}\|^2$. It measures the distance from each data point $\mathbf{z}_t$ to the closest point on the spline $\mathbf{s}_{|\mathbf{z}_t}$, as computed by Schneider [12]. Assuming Gaussian noise and i.i.d. data points, the likelihood is

$$L = \prod_{t=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|\mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t}\|^2}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N e^{-\frac{r^2}{2\sigma^2}}\,. \tag{11}$$

The number of free model parameters $\hat{K}$ obviously depends on the number of control points used to model the spline. Since each control point in our model has 3 parameters and the start and end positions $\mathbf{p}_0^0$ and $\mathbf{p}_M^0$ are given, our model has a complexity of $\hat{K} = 3(M+1) - 4$ parameters. In comparison, the constrained basic 2D splines of order n have $\hat{K} = 2K(M+1) - 6$ parameters, and the unconstrained ones have $\hat{K} = 2K(M+1)$ parameters, with $K = \frac{n+1}{2}$. The BIC is used to choose control points from an initial list using the procedure described in the next section.

C. Optimization procedure

Using a septic spline fit to find curvature extrema, we obtained a set of arc lengths $\mathbb{L} = \{l_j\}$ where control points should potentially be placed. By iteratively applying the following procedure we aim to find a subset $\mathbb{L}^* \subseteq \mathbb{L}$ that minimizes the BIC for the corresponding spline fit.

We obtain $\mathbb{L}^*$ by iteratively removing elements from $\mathbb{L}$. In each step, we tentatively remove each element and perform a non-linear spline fit for the remaining control points. The element whose removal improves the BIC the most is permanently removed. The procedure terminates when no removal improves the BIC. Based on the control point locations in the subset $\mathbb{L}^*$, we refine spline paths as described in Sect. VI. See Fig. 6 for an overview.

VI. Refining Spline Fits

All the least-squares techniques for spline fitting described above exploit a fixed correspondence between data points $\mathbf{z}_t$ and internal parameters u_t. This allows for solving the fit in closed form or with a few steps of non-linear optimization, but also limits the expressiveness of the spline. We therefore propose two additional steps to further refine the spline fits. Firstly, we optimize the correspondences, i.e., the position of control points along the spline, which improves the spline fit in sharp curves as shown in Fig. 3 (D). Secondly, we perform an additional optimization step that relaxes the internal parameter u, which allows the spline to vary its "velocity", i.e., the arc length per internal parameter. This way, the spline can use short tangents to achieve accurate fits even in sharp curves as shown in Fig. 3 (E).

A. Adjusting control point correspondences

In Sect. V-A we defined the set $\mathbb{L}$ of arc lengths along the spline where control points are placed. The method in Sect. V-C prunes $\mathbb{L}$ to a subset $\mathbb{L}^*$. By increasing or decreasing an $l_j \in \mathbb{L}^*$, the corresponding control point moves forwards or backwards along the spline. Thereby, the correspondence between the points on the spline and the training data points is changed. We employ non-linear optimization using the Levenberg-Marquardt algorithm to perform these changes. In every iteration, the optimization adjusts the elements of $\mathbb{L}^*$ and performs new non-linear spline fits to minimize the residual error as shown in Fig. 6.

An example is shown in Fig. 3 (D), where adjusting the location of the upper control point reduces the tension around the control point in the corner visible in (C).

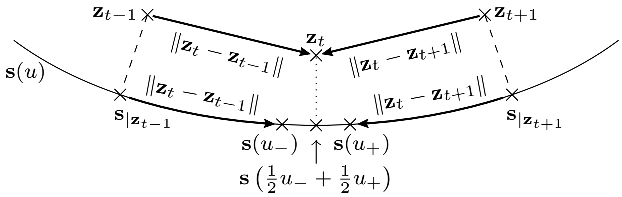

Fig. 7. Our method for computing the fitting error for $\mathbf{z}_t$. We locate the spline points $\mathbf{s}_{|\mathbf{z}_{t-1}}$, $\mathbf{s}_{|\mathbf{z}_{t+1}}$ closest to the data points $\mathbf{z}_{t-1}$, $\mathbf{z}_{t+1}$. Projecting the arc length between the data points onto the spline yields $\mathbf{s}(u_-)$, $\mathbf{s}(u_+)$. Their average is used to determine the error for $\mathbf{z}_t$.

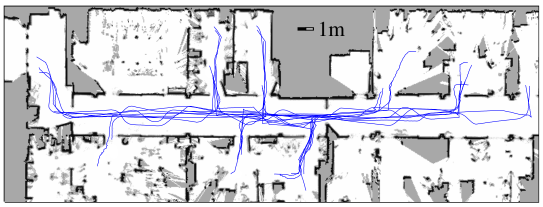

Fig. 8. The 20 reference trajectories recorded for our experiments.

B. Relaxing the internal parameter u

In the previous sections, the splines were fitted using error measures with fixed correspondences between $\mathbf{s}(u_t)$ and $\mathbf{z}_t$. In this way, the "velocity" of a spline segment, i.e., the arc length per u-interval, remains roughly constant. Relaxing this constraint requires extra effort in the computation of the fit errors, but allows for more accurate spline fits.

A simple error measure for least-squares fitting without u correspondences is the closest distance from each data point to the fitted spline. In this case, however, the spline could be close to all data points and still contain deviations and loops without being penalized as shown in Fig. 3 (F). We propose a novel approach that overcomes this problem.

To compute the fitting error for a data point $\mathbf{z}_t$ and a spline $\mathbf{s}(u)$, we consider the neighboring points $\mathbf{z}_{t-1}$ and $\mathbf{z}_{t+1}$ and the spline points $\mathbf{s}_{|\mathbf{z}_{t-1}}$ and $\mathbf{s}_{|\mathbf{z}_{t+1}}$ closest to them, as illustrated in Fig. 7. Starting from $\mathbf{s}_{|\mathbf{z}_{t-1}}$ we move the distance between $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ along the spline to the point denoted by $\mathbf{s}(u_-)$. Similarly, from $\mathbf{s}_{|\mathbf{z}_{t+1}}$ we move to $\mathbf{s}(u_+)$. The points $\mathbf{s}(u_-)$ and $\mathbf{s}(u_+)$ both approximate the point on the spline corresponding to $\mathbf{z}_t$, and we use the average $\mathbf{s}\left(\frac{1}{2}u_- + \frac{1}{2}u_+\right)$ to compute the fit error for $\mathbf{z}_t$.

When performing the non-linear spline fitting using this error measure, the spline is not restricted by the correspondences of u and can be fitted to sharp corners with much higher accuracy as shown in Fig. 3 (E). At the same time, using two neighbors for the closest point search effectively suppresses the degeneration of the fitted spline.

VII. Experiments

To evaluate our approach we recorded 20 trajectories with a real robot driven by joystick in an office building, as shown in Fig. 8. As described in Sect. III-A, the data was pruned with a minimum distance threshold $\tau_l = 0.05\,\mathrm{m}$, which corresponds to the map resolution of the robot localization.

Candidate control point locations were identified by the curvature of a septic spline with 0.5 segments per meter, and thresholded with $\tau_c = 0.1\,\frac{1}{m}$ (see Sect. V-A). These values are appropriate for paths in human environments, but can easily be scaled to miniature or large-size robots.

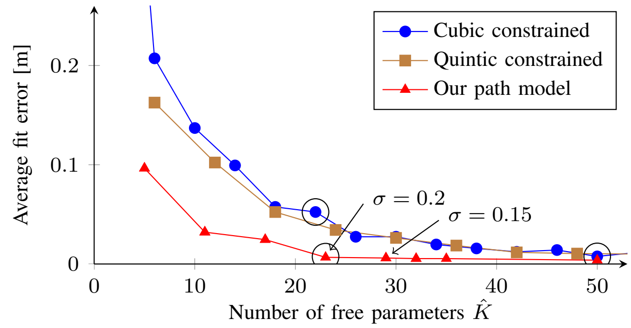

Fig. 9. Residual fitting errors for varying model complexity of the compared approaches. The splines were fitted to the data in Figs. 1 and 2, which also show the fits for the marked combinations (circles).

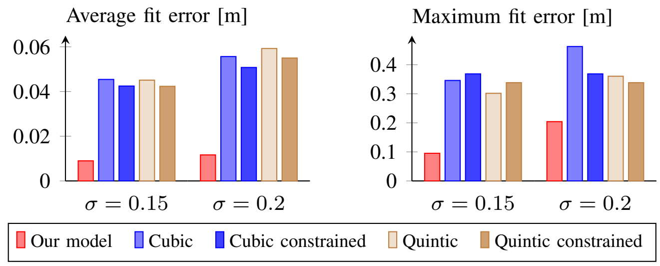

Fig. 10. Average (left) and maximum (right) residual fitting errors for the 20 trajectories used in the experiments. We have selected the linear fits that maximize the BIC for the indicated value of σ.

We fitted our path model to all recorded trajectories. Depending on the value for σ in Eq. (11), our method balances the number of model parameters with the residual fit error. For comparison, we computed constrained and unconstrained linear least-squares fits of cubic and quintic splines (see Sect. III-A). Here, we manually varied the number of segments to achieve different trade-offs.

All of the fitted trajectories were appropriate spline fits and did not suffer from extra loops. To compare the fit quantitatively across trajectories and approaches, we computed the fitting error as average distance from the data points to the fitted path, $E = \frac{1}{N}\sum_{t=0}^{N-1}\|\mathbf{z}_t - \mathbf{s}_{|\mathbf{z}_t}\|$.

Fig. 9 shows errors of different fits for one trajectory. As expected, a higher number of parameters leads to lower errors for all approaches. For our application, $\sigma = 0.15$ or $\sigma = 0.20$ yields a good compromise. In all cases, our approach achieves a lower error for a comparable number of parameters, and needs fewer parameters for comparable errors. The biggest improvements occur in sharp corners, see Fig. 3, (E vs. A). Results for the other trajectories are similar.

Fig. 10 shows the average and maximum fit error per data point over all trajectories. For the baseline approaches, we also computed the BIC for different numbers of control points, and for each trajectory selected the number that maximized the BIC. The fitted curves are therefore the optimal balance between fit error and model complexity for each approach and a given σ. The plot shows that our approach generates substantially lower average and maximum fit errors. The baseline approaches have no significant difference in fit quality for cubic vs. quintic and constrained

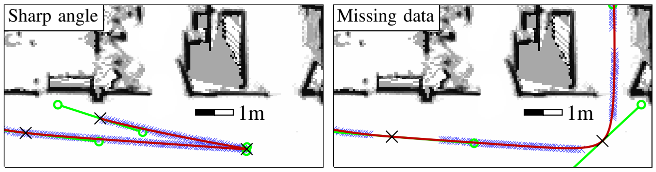

Fig. 11. Our placement of control points and the resulting spline fits are robust to very sharp angles and missing values in the reference data.

vs. unconstrained ones, since the BIC selection uses fewer control points for models with more parameters per point.

Fig. 11 demonstrates the robustness in challenging situations. Neither extreme directional changes nor a considerable amount of missing data deteriorate our spline fits. With increasing values for σ, the fits account for noise in the reference data. Extreme outliers can be filtered out or compensated with robust statistics in the least-squares fits.

Naturally, the computation time for a spline fit depends on the size of the input data. A crucial factor is the number of curvature maxima used in the combinatorial control point selection, see Sect. V. Since relaxed fitting (Sect. VI-B) requires substantially more time to compute residual errors, it is only used in a post-processing step, see Fig. 6.

VIII. Applications

As shown in the experiments, our approach generates accurate spline fits with a small number of parameters. This section presents several applications based on this property.

After fitting the path model, a robot can follow the path using ad-hoc velocities set by a controller. One can also compute a velocity profile for the path using the method by Lau *et al.* [10]. This minimizes the traversal time by maximizing the translational velocity, while obeying a set of constraints, e.g., maximum speed and acceleration of the hardware platform, or a bound on the centripetal force. In this case, the path shape remains unchanged. Additionally, one can also employ the optimization procedure in [10] to optimize the path shape as well, e.g., with user-specified bounds on the allowed deviation from the reference path. The small number of parameters in the fitted trajectory makes this problem computationally feasible. In this way, a suboptimal shape of the reference trajectory can be optimized to yield faster travel times as well, while retaining the topology of the path and avoiding collisions with mapped obstacles.

When combining our spline fitting method with the autonomous trajectory generation in [10], the robot can be programmed by demonstration to follow a path. If the path is blocked during execution, it can leave the path to circumvent the obstacle and then return to the assigned path.

This approach can also be used to autonomously plan a trajectory that leads back to the reference path, e.g., to reduce accumulated errors when using the odometry for trajectory execution, or to recover after detecting a localization failure. Therefore, we have to create a new spline segment $\mathbf{s}_{\text{new}}$ that connects the current robot pose to the existing spline $\mathbf{s}(u)$. To achieve a smooth join at a given point $\mathbf{s}(u_d)$ on $\mathbf{s}(u)$, we subdivide the segment of $\mathbf{s}(u)$ that is active at u_d by inserting an extra control point $\mathbf{p}_d$ at u_d. Its parameters $\mathbf{p}_d^k$ are determined by the old segment and its derivatives at that point, after rescaling u to account for the new length of the subdivided segment.

Finally, the new segment $\mathbf{s}_{\text{new}}$ and the location of the join point u_d can be optimized. This is done using the time of travel as cost function with an additional penalty for prolonged deviation from the target path. As the segment $\mathbf{s}_{\text{new}}$ should join the divided segment of $\mathbf{s}(u)$ with continuous curvature, the tangent orientation $\mathbf{p}_d^1$ of $\mathbf{p}_d$ does not use our heuristic but is fixed to the one given by $\mathbf{s}'(u_d)$.

IX. Conclusion

We presented an approach to robustly fit parametric mobile robot paths to reference trajectories recorded by a user. Our method uses a specific path model that needs fewer parameters than standard approaches to achieve similar approximation results. We employ the Bayesian Information Criterion in the optimization procedure to calculate the best trade-off between model complexity and accuracy. The experiments carried out on real-world data show that our approach clearly outperforms basic spline fitting methods.

We believe that the presented approach allows for intuitive and flexible teaching of robot paths and supports several applications: fitted paths can be augmented with time-efficient velocity profiles and further optimized to minimize the time of travel. In addition, our method can be combined with trajectory optimizers, e.g., to avoid unexpected obstacles.

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–19, 2008.

[2] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation: an approach based on Hidden Markov Model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44–54, 2010.

[3] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47, pp. 69–77, 2004.

[4] J. Kolter and A. Ng, "Task-space trajectories via cubic spline optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009, pp. 1675–1682.

[5] C. Lee, "A phase space spline smoother for fitting trajectories," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 346–356, Feb. 2004.

[6] P. Baiget, E. Sommerlade, I. Reid, and J. Gonzáles, "Finding prototypes to estimate trajectory development in outdoor scenarios," in *Intl. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, September 2008.

[7] W. Wang, H. Pottmann, and Y. Liu, "Fitting b-spline curves to point clouds by curvature-based squared distance minimization," *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 214–238, April 2006.

[8] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[9] S. Macfarlane and E. Croft, "Design of jerk bounded trajectories for online industrial robot applications," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, 2001, pp. 979–984.

[10] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), St. Louis, MO, USA*, 2009.

[11] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard, "Online generation of kinodynamic trajectories for non-circular omnidirectional robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, 2011.

[12] P. J. Schneider, "Solving the nearest-point-on-curve problem," in *Graphics Gems*, A. S. Glassner, Ed. Academic Press, Inc., 1990.