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 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.

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].

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).
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 \( \tilde{s}(u) \) 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 \( p_i \) at its start \((p_{s, i})\) and end \((p_{e, i})\). Each \( p_i \) has \( K + \frac{3}{2} \) parameters \( p_{s, i}^k, k = 0, \ldots, K - 1 \). The \( p_i^k \) are vectors with one component per spline dimension, and specify the \( k \)-th derivative of \( \tilde{s}(u) \) at the start \((u = 0)\) and end \((u = 1)\) of the segment. \( \tilde{s}(u) \) is then given by the linear combination

\[
\tilde{s}(u) = \sum_{k=0}^{K-1} h^k(u) \cdot p^k + h^k(u) \cdot p^k_e,
\]

where \( h^k(u) \) and \( h^k_e(u) \) are polynomials called Hermite basis functions. They are obtained by solving

\[
\tilde{s}^{(k)}(0) = p^k, \quad \tilde{s}^{(k)}(1) = p^k_e, \quad k = 0, \ldots, K - 1,
\]

where \( \tilde{s}^{(k)} \) is the \( k \)-th derivative of the polynomial \( \tilde{s} \). By factoring out the \( p_s^k, p_e^k \) we obtain the basis functions for cubic, quintic and septic splines shown in the following table.

<table>
<thead>
<tr>
<th>Cubic (K=2)</th>
<th>Quintic (K=3)</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_0^1 ) ( 2u^3 - 3u^2 + 1 )</td>
<td>( p_0^1 ) ( 3u^4 - 4u^3 + 1 )</td>
<td>( p_1^0 )</td>
</tr>
<tr>
<td>( h_1^1 ) ( u^3 - 2u^2 + u )</td>
<td>( h_2^1 ) ( 3u^4 - 4u^3 + u )</td>
<td>( p_2^0 )</td>
</tr>
<tr>
<td>( h_2^1 ) ( -2u^3 + 3u^2 )</td>
<td>( h_3^1 ) ( 2u^4 - 3u^3 + u )</td>
<td>( p_3^0 )</td>
</tr>
<tr>
<td>( h_3^1 ) ( u^3 - u^2 )</td>
<td>( h_4^1 ) ( -2u^4 + 2u^3 - 4u^2 + u )</td>
<td>( p_4^0 )</td>
</tr>
<tr>
<td>( h_4^1 ) ( -2u^3 + 3u^2 )</td>
<td>( h_5^1 ) ( 2u^4 - 3u^3 + 2u^2 - 4u + 1 )</td>
<td>( p_5^0 )</td>
</tr>
</tbody>
</table>

Typically, a spline curve is a concatenation of multiple spline segments. When joining \( M \) segments \( \tilde{s}_i, i \in \{0, \ldots, M-1\} \), the resulting curve \( s(u) \) is defined over \([0, M]\) and given by \( s(u) = \tilde{s}_i(u - i) \). Here, \( \tilde{s}_i \), with \( i = \lfloor u \rfloor \) is the “active” segment for a certain \( u \), and specified by \( p_{s,i}^k = p_s^k \) and \( p_{e,i}^k = p_e^k \). 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 \( z(t) \), we want to find an accurate parametric approximation with as few parameters as possible. \( z(t) \) is given by the robot position \( z_t = (x_t, y_t) \) at each discrete time step \( t = 0, \ldots, N-1 \). We compute the cumulative length of the piecewise linear path given by \( z(t) \) as \( l_t = \sum_{i=1}^{t} \|z_t - z_{t-1}\| \). We assume that the \( z_t \) have been pruned to have a minimum distance \( l_t - l_{t-1} > \tau_l \) for all \( t \). To approximate \( z(t) \) with a spline, we assign each \( z_t \) a corresponding \( u_t \) by linear interpolation with respect to the arc length, \( u_t = M \cdot (l_t/l_{N-1}) \).
parameter vector \( \mathbf{p} \), and replaced by \( e_0 \) and \( e_M \). The corresponding coefficients in \( X \) are \( \delta_{i=0} = h^4_i(u_i - i) \cdot s'(0) \) and \( \delta_{i=M-1} = h^4_i(u_i - i) \cdot s'(M) \), 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

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

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:

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

where \( s''_i \) 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.

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 \( \mathbf{p}^+ = (p_1^0 \cdots p_{M-1}^0 \ e_0 \cdots e_M)^T \) 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

\[
\mathbf{p}^+ = \arg \min_{\mathbf{p}^+} \| \mathbf{z} - X \cdot f(\mathbf{p}^+) + \mathbf{b} \|.\tag{8}
\]

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 \( \mathbf{p}^0_i \) and \( e_i \). The \( e_i \) are determined from the \( \mathbf{p}^1_i \) 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 \( s_c(u) \) 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

\[
c(u) = \frac{s'_c \times s''_c}{\|s'_c\|^3}, \quad c'(u) = \frac{s'_c \times s''_c}{\|s'_c\|^3} - \frac{3 (s'_c \times s''_c)(s'_c \cdot s''_c)}{\|s'_c\|^5},
\]
where \( a \times b = a_x b_y - a_y b_x \) for the \( x \) and \( y \) components of the 2D spline \( s(u) \) and its derivatives. Again, we dropped the dependency of \( s_u(u) \) on \( u \) for readability.

Since \( c'(u) \) depends on the third derivative \( s''_u(u) \), we use a septic spline for \( s(u) \), for which \( s''_u(u) \) 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 \( z_t \). The start and end points 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 on the spline to these \( l_j \) by associating them with integer \( u \) values. Then, the spline parameter \( u_j \) for a data point \( z_t \) at arc length \( l_j \) between two anchored control points with indices \( j, j+1 \) and arc lengths \( l_j, l_{j+1} \) is given by the interpolation

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

### 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),

\[
 \text{BIC} = -2 \log L + \hat{K} \log N, \tag{10}
\]

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 \( r^2 = \sum_{t=0}^{N-1} ||z_t - s_{|k|}||^2 \). It measures the distance from each data point \( z_t \) to the closest point on the spline \( s_{|k|} \), as computed by Schneider [12]. Assuming Gaussian noise and i.i.d. data points, the likelihood is

\[
 L = \prod_{t}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{||z_t - s_{|k|}||^2}{2\sigma^2}} = \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^N e^{-\frac{2}{\sigma^2}}. \tag{11}
\]

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 \( p_0^0 \) and \( p_M^0 \) 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 \( L = \{l_j\} \) where control points should potentially be placed. By iteratively applying the following procedure we aim to find a subset \( L^* \subseteq L \) that minimizes the BIC for the corresponding spline fit.

We obtain \( L^* \) by iteratively removing elements from \( L \). 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 \( L^* \), 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 \( z_t \) 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 \( L \) of arc lengths along the spline where control points are placed. The method in Sect. V-C prunes \( L \) to a subset \( L^* \). By increasing or decreasing an \( l_j \in L^* \), 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 \( L^* \) 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).
values are appropriate for paths in human environments, but can easily be scaled to miniature or large-size robots.

We fitted our path model to all recorded trajectories. Depending on the value for $\sigma$ 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, $E = \frac{1}{N} \sum_{i=0}^{N-1} \| z_t - s(u) \|$. 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 for different fits 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 $\sigma$. 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 approaches.

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 $\sigma$.

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 $\tau_1 = 0.05 \text{m}$, 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
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.

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 \( s_{\text{new}} \) 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 \( s_{\text{new}} \) should join the divided segment of \( s(u) \) with continuous curvature, the tangent orientation \( p_d^T \) of \( p_d \) does not use our heuristic but is fixed to the one given by \( s'(u_d) \).

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


