Cartesian Spline Interpolation for Industrial Robots

Thomas Horsch *(Heusenstamm, Germany)*
Bert Jüttler *(TU Darmstadt, Germany)*

University of Technology, Darmstadt
Department of Mathematics
Schloßgartenstr. 7
64289 Darmstadt, Germany
Phone: +49 6151 16 2789, Fax: +49 6151 16 2131
horsch@t-online.de, juettler@mathematik.tu-darmstadt.de

We describe an algorithm for interpolation of positions by a rational spline motion. A reparameterization of the resulting motion is applied in order to achieve the desired distribution of the velocity. For the ease of presentation we discuss trapezoidal velocity profiles, i.e., piecewise constant and linear velocity distribution. The method can be generalized to more general velocity profiles too. The whole spline scheme possesses some special features which make it a suitable tool for the control of industrial robots.

Keywords: Rational spline motions, Cartesian spline interpolation, industrial robots

Introduction

By using spatial rational spline motions it is possible to apply many of the powerful methods of Computer Aided Geometric Design (as Bézier or B-spline techniques) in order to solve problems out of the field of Kinematics and Robotics. Rational spline motions are characterized by the property that the trajectories of the points of the moving object are rational spline curves, i.e., the trajectories are NURBS (Non–Uniform Rational B–Spline) curves (see the textbook by Hoschek and Lasser[1]). Due to their geometric flexibility and computational efficiency, NURBS curves and surfaces became an industrial standard (STEP) for the data exchange between CAD systems. Therefore rational spline motions seem to be the appropriate tool for the mathematical description of motions regarding industrial applications.

In the near future, the programming and control of robots will increasingly make use of CAD data. In general, this data will specify the desired trajectory of the Tool–Center point (TCP) only, but not the orientation of the end–effector. Great attention should be paid to the direct integration of CAD data into the robot control (as far as possible). Rational spline motions seem to be well suited for dealing with this task. On the other hand, we are able to develop methods for simplified programming of complex robot motions based on rational spline motions using the traditional "teach–in programming."

Most commercial robot controllers use either piecewise linear or circular interpolation schemes, see References 2 or 3. As a major advantage of those schemes, the exact arc–length of the trajectory of the TCP is known and it can be used for achieving the desired velocity distribution. In order to overcome the limitations of linear and circular interpolation, Wang and Yang[4] developed a scheme for nearly arc–length parameterized quintic spline curves. Farouki and Shah[5] suggested another approach which is based on so–called Pythagorean–hodograph curves. Such curves possess a polynomial arc–length function. For both schemes, however, the interpolant has to be computed off–line as the required CPU times are too large.

In contrast, our spline scheme should be used as a real–time interpreter of the robot program. In addition, our spline scheme deals simultaneously with the translational and the rotational part of the given positions. The methods described below have been developed as a part of a new controller for industrial robots of various geometries.

At first, we present an algorithm which generates a rational spline motion from a sequence of given positions, e.g. of teach points. Corner smoothing (which is necessary for certain applications) can be handled by introducing artificial teach points. Circular arcs can also be represented. This ensures the downward–compatibility of the new controller. By using spline motions we get a unified data structure for the linear, circular and spline interpolation schemes. The spline algorithm works completely local, i.e., the construction of each segment is only based on a small number of neighboured positions.

In the second part we discuss the discretization of the spline motion in order to perform the desired motion of the robot manipulator. By applying a reparameterization we construct motions with a piecewise constant or linear velocity distribution. For instance, such motions are required for welding applications. The scheme can be generalized to more general velocity profiles (e.g. 7–segment C^1 profiles). For the ease of presentation we discuss only trapezoidal profiles.

It is shown that the arc length of spline curves can be approximated with sufficient accuracy in real time. This enables us to benefit from the increased flexibility and continuity of spline motions. The paper concludes with some forthcoming questions concerning (amongst others) the direct integration of CAD data.

In this paper we are not considering robot dynamics. For most CP (Cartesian path) applications (such as arc welding, deburring, clueing) robot dynamics is of minor importance, since the robot velocity is limited by the process. Taking into account robot dynamics would not affect our approach very much: One simply would need to substitute the trapezoidal velocity profiles (which we use) by the profiles which have been calculated including the dynamics.

Rational motions

In addition to the underlying world xyz-coordinates (with respect to a *fixed* frame), the moving end-effector of the robot is equipped with another Cartesian $\widehat{x}\widehat{y}\widehat{z}$-coordinate frame (called the *moving* frame). The origin of the moving frame is chosen at the Tool Center Point (TCP). The motion of the end effector is represented by the trajectory $\mathbf{m}(t) = (\, m_1(t)\ m_2(t)\ m_3(t)\,)^\top$ of the TCP and by the proper orthogonal 3×3 matrix $R(t)$ which describes the orientation of the moving frame. The parameter t can be considered as the time. During the motion, any point $\widehat{\mathbf{p}}$ on the end-effector (which is represented by its coordinates $(\,\widehat{x}_p\;\widehat{y}_p\;\widehat{z}_p\,)^\top$ with respect to the moving frame) runs along the trajectory

$$\mathbf{p}(t) = \mathbf{m}(t) + R(t)\cdot\widehat{\mathbf{p}}. \tag{1}$$

This equation describes the coordinate transformation $\widehat{\mathbf{p}} \mapsto \mathbf{p}(t)$ from the moving coordinate system of the end-effector into world coordinates. If the trajectories of all points $\widehat{\mathbf{p}}$ are (piecewise) rational curves, then the Euclidean motion of the end-effector is said to be a (piecewise) rational motion. Piecewise rational (i.e., NURBS) curves are of fundamental importance in Computer Aided Geometric Design. They are usually described in Bézier- or B-spline form, see Hoschek and Lasser[1]. We present a simple construction of spatial rational motions. Let

$$\mathbf{m}(t) = \frac{1}{u_0}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{2}$$

and

$$R(t)=\frac{1}{D}\begin{pmatrix} d_0^2+d_1^2 -d_2^2-d_3^2 & -2\,d_0d_3 +2\,d_1d_2 & 2\,d_0d_2 +2\,d_1d_3 \\ 2\,d_0d_3 +2\,d_1d_2 & d_0^2-d_1^2 +d_2^2-d_3^2 & -2\,d_0d_1 +2\,d_2d_3 \\ -2\,d_0d_2 +2\,d_1d_3 & 2\,d_0d_1 +2\,d_2d_3 & d_0^2-d_1^2 -d_2^2+{d_3}^2 \end{pmatrix} \tag{3}$$

with $D = d_0^2+d_1^2+d_2^2+d_3^2$, where the eight functions $u_0{=}u_0(t),\dots,u_3{=}u_3(t)$ and $d_0{=}d_0(t),\dots,d_3{=}d_3(t)$ are polynomials in t. Then, the transformation (1) between moving and world coordinates describes a rational motion. It can be shown that all rational motions result from this construction. For more information on rational motions we refer to the survey by Röschel[6].

The denominator $u_0(t)$ of the TCP trajectory has been introduced only in order to be compatible with NURBS curves. Throughout the construction of the interpolating spline motion we will choose $u_0(t)\equiv 1$. By using a non-constant denominator function $u_0(t)$, it is possible to construct exact representations of circular arcs and other conic sections, see Reference 1, Chapter 4.1.4.

The *velocity* of the TCP with respect to t results out of

$$\vec{\mathbf{v}}(t) = -\frac{{u_0}'}{u_0^2}\vec{\mathbf{u}} + \frac{1}{u_0}\vec{\mathbf{u}}' \tag{4}$$

in which $\vec{\mathbf{u}}(t) = (\,u_1(t)\;u_2(t)\;u_3(t)\,)^\top$. The prime $'$ denotes the derivative with respect to the (time) parameter t.

Note that Eq. (3) is the classical representation of a rotation matrix $R(t)$ with the help of *Euler parameters*, see e. g. Bottema and Roth[7]. As an abbreviation, the four Euler parameters of the rotation matrix $R(t)$ are collected in the four-dimensional *"Euler vector"*

$$\widetilde{\mathbf{d}}(t) = (\,d_0(t)\;d_1(t)\;d_2(t)\;d_3(t)\,)^\top\,. \tag{5}$$

The first derivative

$$\widetilde{\mathbf{d}}'(t) = (\,{d_0}'(t)\;{d_1}'(t)\;{d_2}'(t)\;{d_3}'(t)\,)^\top$$

of this vector with respect to t will be called the *"Euler velocity"*. The corresponding *angular velocity* of the end-effector is equal to

$$\vec{\omega}\,(t) = \frac{2}{\widetilde{\mathbf{d}}^\top\widetilde{\mathbf{d}}}\left(\vec{\mathbf{d}}\times\vec{\mathbf{d}}' - {d_0}'\vec{\mathbf{d}} + d_0\vec{\mathbf{d}}'\right), \tag{6}$$

with $\vec{\mathbf{d}}(t) = (\,d_1(t)\;d_2(t)\;d_3(t)\,)^\top$.

We do not assume that the Euler vectors $\widetilde{\mathbf{d}}(t)$ have always been normalized. (The Euler parameters $\widetilde{\mathbf{d}}$ are said to be *normalized* if $\widetilde{\mathbf{d}}^\top\cdot\widetilde{\mathbf{d}} = d_0^2+d_1^2+d_2^2+d_3^2 = 1$. Normalized Euler parameters are identical with the components of the unit quaternion which describes the rotation $R(t)$, see References 6 and 7.) Using normalized Euler parameters as proposed by

Johnstone and Williams[8] would cause a doubling of the polynomial degrees which are required to solve the interpolation problem.

Later on we will apply a reparameterization $t = t(\tau)$ of the motion in order to realize the desired distribution of velocity. After that, the original parameter t of the motion will not be the time in general.

The interpolation problem

A sequence of $N+1$ positions $\mathrm{Pos}_0, \dots, \mathrm{Pos}_N$ of the robot end-effector is assumed to be given. This sequence is normally generated by "teach–in" programming. Each position Pos_i is described by the coordinates $\mathbf{r}_i = (\, x_i \; y_i \; z_i \,)^\top$ of TCP and the rotation matrix S_i. At the i-th position Pos_i the mapping $\widehat{\mathbf{p}} \mapsto \mathbf{p}$ with

$$\mathbf{p} = \mathbf{r}_i + S_i \cdot \widehat{\mathbf{p}} \tag{7}$$

transforms the $\widehat{x}\widehat{y}\widehat{z}$-coordinates of the end-effector into world coordinates, cf. (1). The rotation matrix S_i is described by its *normalized* Euler parameters $\widetilde{\mathbf{q}}_i = (\, q_{i,0} \; q_{i,1} \; q_{i,2} \; q_{i,3} \,)^\top$. Note that the two normalized Euler vectors $\widetilde{\mathbf{q}}_i$ and $-\widetilde{\mathbf{q}}_i$ correspond to the same rotation S_i ! We adjust the signs of the Euler parameters in such a way, that the inequalities

$$\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1} = \sum_{j=0}^{3} q_{i,j}\, q_{i+1,j} \geq 0 \quad (i = 0, \dots, N-1) \tag{8}$$

are fulfilled. Then the angles between adjacent Euler vectors $\widetilde{\mathbf{q}}_i$ in $\mathbb{R}^4$ become as small as possible. Taking the first Euler parameters $\widetilde{\mathbf{q}}_0$ and using condition (8), the signs of the remaining vectors $\widetilde{\mathbf{q}}_1, \dots, \widetilde{\mathbf{q}}_N$ are uniquely determined, provided that $\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1} \neq 0$, i.e. that the angle of the rotation $S_i{}^{-1} \cdot S_{i+1}$ from Pos_i to Pos_{i+1} is always less than π. (The angle of this rotation is equal to $2 \cdot \arccos(\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1})$, see Reference 7.)

In addition to the positions Pos_i, a strictly monotonic sequence of real parameter values $t_0 < t_1 < \dots < t_N$ is assumed to be given. For the value $t = t_i$ of the motion parameter, the interpolating motion will interpolate the given position Pos_i. If the parameters t_i are yet unknown, then they have to be estimated out of the given data. In our implementation we chose $t_0 = 0$ and

$$t_{i+1} = t_i + \max \{\, \Delta_{\min},\, \gamma \cdot \|\mathbf{r}_{i+1} - \mathbf{r}_i\|, \\ 2\delta \cdot \arccos(\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1}) \,\}$$

for $i = 0, \dots, N-1$. (The symbol $\|\vec{\mathbf{x}}\|$ denotes the norm $(\vec{\mathbf{x}}^\top \vec{\mathbf{x}})^{\frac{1}{2}}$ of the (Euler) vector $\vec{\mathbf{x}}$.) By using this formula the difference of adjacent parameter values is related to the distance between neighboured positions. Note that also the difference of the orientations is taken into account. The constants γ, δ and $\Delta_{\min}$ have to be specified by the user. The positive real numbers γ and δ control the influence of the rotational and the translational part to the choice of the parameter values t_i, whereas $\Delta_{\min} > 0$ is the minimal allowed difference of the adjacent parameter values. The ratio $\frac{\gamma}{\delta}$ should correspond to the maximal acceleration of the TCP to the maximal angular acceleration of the robot.

We will construct an interpolating rational spline motion. The construction will use the sufficient interpolation conditions

$$u_0(t_i) = 1, \quad \begin{pmatrix} u_1(t_i) \\ u_2(t_i) \\ u_3(t_i) \end{pmatrix} = \vec{\mathbf{r}}_i \tag{9}$$

for the translational part and

$$\widetilde{\mathbf{d}}(t_i) = \widetilde{\mathbf{q}}_i \tag{10}$$

for the rotational part, $i = 0, \dots, N$. These conditions are not necessary, as they introduce some normalizations (for $u_0(t)$ and for the norm of the Euler vectors $\widetilde{\mathbf{d}}(t_i)$). The general motion interpolation problem and the use of the resulting additional degrees of freedom is discussed in Reference 9.

Based on cubic spline functions we will derive an interpolation scheme which possesses the following important features:

- Each segment of the spline motion is obtained by a *local* construction which is based on a finite number of neighboured positions. All computations can be done in *real time*, i.e. during the movement of the robot manipulator. Therefore, the construction can be used as an *interpreter scheme* of the given positions.
- The construction yields a rational C^1-spline motion. Hence, the velocity of the TCP and the angular velocity of the end-effector are always continuous.
- Circular arcs and line segments are available as segments of the trajectory of the TCP. This ensures compatibility with recently used algorithms.

These features will make the interpolation scheme a suitable tool for the control of robots.

Construction of the spline motion

From the given $N+1$ positions $\mathrm{Pos}_0, \ldots, \mathrm{Pos}_N$ of the end-effector we construct an interpolating rational spline motion. The construction of the spline motion consists of three steps:

1. Estimating the velocities. At each given position Pos_i we estimate the velocity $\vec{\mathbf{v}}_i = \vec{\mathbf{v}}(t_i)$ of the TCP and the Euler velocity $\widetilde{\mathbf{e}}_i = \widetilde{\mathbf{d}}'(t_i)$ of the moving end-effector with respect to the parameter t. At first, we consider the two neighbouring positions Pos_i and Pos_{i+1} with the parameters $t = t_i$ and $t = t_{i+1}$. The velocity of the uniform translation of the TCP from $\mathbf{r}_i$ to $\mathbf{r}_{i+1}$ is equal to

$$\vec{\mathbf{v}}_i^{(+)} = \vec{\mathbf{v}}_{i+1}^{(-)} = \frac{1}{\Delta t_i}\left(\mathbf{r}_{i+1} - \mathbf{r}_i\right) \tag{11}$$

($i = 0, \ldots, N-1$) with $\Delta t_i = t_{i+1} - t_i$. Similarly, the Euler velocities at $t = t_i$ and $t = t_{i+1}$ of the uniform rotation $S_i^{-1} \cdot S_{i+1}$ from Pos_i to Pos_{i+1} are

$$\begin{aligned} \widetilde{\mathbf{e}}_i^{(+)} &= \frac{\phi_{i,i+1}}{\Delta t_i \cdot \sin\phi_{i,i+1}} \left(\widetilde{\mathbf{q}}_{i+1} - (\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1}) \cdot \widetilde{\mathbf{q}}_i\right) \\ &\text{and} \\ \widetilde{\mathbf{e}}_{i+1}^{(-)} &= \frac{\phi_{i,i+1}}{\Delta t_i \cdot \sin\phi_{i,i+1}} \left((\widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1}) \cdot \widetilde{\mathbf{q}}_{i+1} - \widetilde{\mathbf{q}}_i\right), \end{aligned} \tag{12}$$

($i = 0, \ldots, N-1$) respectively, where $\phi_{i,i+1} = \arccos \widetilde{\mathbf{q}}_i^\top \widetilde{\mathbf{q}}_{i+1}$.

A couple of different methods is available for estimating the velocities at the given positions, e.g. the Bessel or the Akima scheme, see Reference 1. But none of them possesses the following additional feature which arises from the interpolation of positions generated by the "teach–in" programming:

- If the TCPs $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ of the adjacent positions are identical but different from $\mathbf{r}_{i-1}$, then the estimate for $\vec{\mathbf{v}}_i$ is expected to be the null vector and the trajectory of the TCP has to have a sharp corner at that point. A similar feature is required for the estimate of the angular velocity.

With the help of these features the user can achieve simple geometric forms like polygonal lines by teaching double positions.

It is obvious to use the averages

$$\frac{1}{2}(\vec{\mathbf{v}}_i^{(-)} + \vec{\mathbf{v}}_i^{(+)}) \quad \text{and} \quad \frac{1}{2}(\widetilde{\mathbf{e}}_i^{(-)} + \widetilde{\mathbf{e}}_i^{(+)})$$

(cf. Figure 1) as estimates of the velocity $\vec{\mathbf{v}}_i$ and of the Euler velocity $\widetilde{\mathbf{e}}_i$. But this choice is not going to

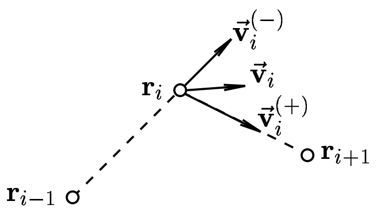

Figure 1: Estimating the velocities

meet the above–mentioned requirement. So we use *scaled averages:*

$$\begin{aligned} \vec{\mathbf{v}}_i &= \tfrac{1}{2}\rho_i\,(\vec{\mathbf{v}}_i^{(-)} + \vec{\mathbf{v}}_i^{(+)}) \quad \text{and} \\ \widetilde{\mathbf{e}}_i &= \tfrac{1}{2}\sigma_i\,(\widetilde{\mathbf{e}}_i^{(-)} + \widetilde{\mathbf{e}}_i^{(+)}) \end{aligned} \tag{13}$$

with

$$\rho_i = \min\left\{1,\ \frac{\alpha\cdot\min\{\|\vec{\mathbf{v}}_i^{(-)}\|,\ \|\vec{\mathbf{v}}_i^{(+)}\|\}}{\|\frac{1}{2}(\vec{\mathbf{v}}_i^{(-)} + \vec{\mathbf{v}}_i^{(+)})\|}\right\} \quad \text{and}$$

$$\sigma_i = \min\left\{1,\ \frac{\alpha\cdot\min\{\|\widetilde{\mathbf{e}}_i^{(-)}\|,\ \|\widetilde{\mathbf{e}}_i^{(+)}\|\}}{\|\frac{1}{2}(\widetilde{\mathbf{e}}_i^{(-)} + \widetilde{\mathbf{e}}_i^{(+)})\|}\right\}$$

(i=1,…,N−1). The velocities in the first and last position are set to zero:

$$\vec{\mathbf{v}}_0 = \vec{\mathbf{v}}_N = \vec{\mathbf{0}} \text{ and } \widetilde{\mathbf{e}}_0 = \widetilde{\mathbf{e}}_N = \widetilde{\mathbf{0}}. \tag{14}$$

The positive real parameter $\alpha \in \mathbb{R}$ acts as a *tension parameter* of the spline motion. For small values of α the TCP trajectory is pulled to the polygon $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N$ (cf. Figure 2). The Euler vectors possess a similar property. A reasonable value of this parameter is $\alpha = 1.2$.

2. Constructing the spline segments. The i-th spline segment describes the motion of the end-effector from i-th position Pos$_i$ to i+1-st position Pos$_{i+1}$ (i = 0,…,N−1). At all parameter values t satisfying $t_i \le t < t_{i+1}$ the trajectory of the TCP results out of (2) with $u_0(t) \equiv 1$ and

$$\begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} = \sum_{j=0}^{3} B_j^3\left(\frac{t-t_i}{\Delta t_i}\right)\cdot \mathbf{b}_j{}^{(i)} \tag{15}$$

with the four 3-vector-valued coefficients

$$\begin{aligned} &\mathbf{b}_0{}^{(i)} = \mathbf{r}_i, && \mathbf{b}_1{}^{(i)} = \mathbf{r}_i + \tfrac{1}{3}\Delta t_i\cdot\vec{\mathbf{v}}_i, \\ &\mathbf{b}_2{}^{(i)} = \mathbf{r}_{i+1} - \tfrac{1}{3}\Delta t_i\cdot\vec{\mathbf{v}}_{i+1} \text{ and } && \mathbf{b}_3{}^{(i)} = \mathbf{r}_{i+1}, \end{aligned} \tag{16}$$

where the $B_j^3(s) = \binom{3}{j}\cdot s^j(1-s)^{3-j}$ denote the cubic Bernstein polynomials ($j = 0, 1, 2, 3$). Similarly, we obtain the rotation matrix $R(t)$ out of (3) with

$$\widetilde{\mathbf{d}}(t) = \sum_{j=0}^{3} B_j^3\left(\frac{t-t_i}{\Delta t_i}\right)\cdot\widetilde{\mathbf{f}}_j^{(i)} \tag{17}$$

with the four 4-vector-valued coefficients

$$\begin{aligned} &\widetilde{\mathbf{f}}_0^{(i)} = \widetilde{\mathbf{q}}_i, && \widetilde{\mathbf{f}}_1^{(i)} = \widetilde{\mathbf{q}}_i + \tfrac{1}{3}\Delta t_i\cdot\widetilde{\mathbf{e}}_i, \\ &\widetilde{\mathbf{f}}_2^{(i)} = \widetilde{\mathbf{q}}_{i+1} - \tfrac{1}{3}\Delta t_i\cdot\widetilde{\mathbf{e}}_{i+1} \text{ and } && \widetilde{\mathbf{f}}_3^{(i)} = \widetilde{\mathbf{q}}_{i+1}. \end{aligned} \tag{18}$$

The i-th spline segment satisfies the interpolation conditions (9) and (10) for $t = t_i$ and $t = t_{i+1}$. Moreover, it interpolates at Pos$_i$ and at Pos$_{i+1}$ the estimated velocities and Euler velocities $\vec{\mathbf{v}}_i$, $\vec{\mathbf{v}}_{i+1}$, $\widetilde{\mathbf{e}}_i$ and $\widetilde{\mathbf{e}}_{i+1}$, see (13).

The Euler velocities $\widetilde{\mathbf{e}}_i$ fulfill $\widetilde{\mathbf{q}}_i^\top\widetilde{\mathbf{e}}_i = 0$, i.e., they are tangential to the unit sphere of $\mathbb{R}^4$ at $\widetilde{\mathbf{q}}_i$ ($i = 0, \ldots, N$). In addition we have $\widetilde{\mathbf{q}}_i^\top\widetilde{\mathbf{q}}_i = 1$. Thus, the interpolating Euler vectors $\widetilde{\mathbf{d}}(t)$ are almost normalized everywhere. This ensures a relatively uniform distribution of the angular velocity of the interpolating motion.

An example of a rational spline motions obtained by the above construction has been drawn in Figure 2a. Six positions with equidistant parameters t_i are interpolated. The TCPs of positions 3 and 4 are identical. The TCP trajectory of the spline segment $t_3 \le t \le t_4$ is degenerated to one point. Similarly, the rotation matrices of positions 2 and 3 are identical. The rotational part $R(t)$ of the segment $t_2 \le t \le t_3$ is constant.

The interpolating spline motion has been computed with the value $\alpha = 1.2$ of the tension parame-

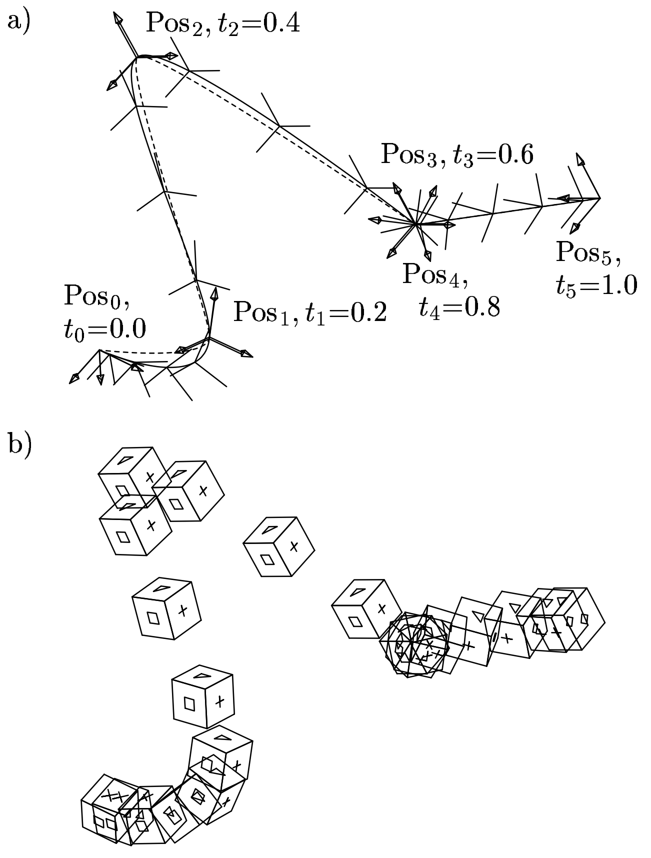

Figure 2: A rational spline motion which interpolates six given positions

ter. In contrast to this, the dashed curve is the trajectory of the TCP which is obtained with $\alpha = 0.3$. This curve is a lot closer to the polygon $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_5$ than the original trajectory of the TCP.

In Figure 2b the interpolating motion is illustrated by some positions of the moving unit cube the faces of which have been marked by crosses, squares, and triangles.

3. Building the B-spline representation. The TCP trajectory $\mathbf{m}(t)$ and the Euler vector $\widetilde{\mathbf{d}}(t)$ (15) and (17) are piecewise cubic polynomial C^1 vector–valued functions. Hence, it is possible to represent them as B-spline functions. Consider the $2N+2$ B-spline basis functions $N_{0,4}(t), \ldots, N_{2N+1,4}(t)$ of order four defined over the knot sequence

$$(\underbrace{t_0, .., t_0}_{\text{4 fold}}, \underbrace{t_1, t_1}_{\text{double}}, \underbrace{t_2, t_2}_{\text{double}}, .., \underbrace{t_{N-1}, t_{N-1}}_{\text{double}}, \underbrace{t_N, .., t_N}_{\text{4 fold}}). \tag{19}$$

For more information concerning B-spline functions see Reference 1 or a similar textbook on spline functions. The trajectory of the TCP has the B-spline representation

$$u_0(t) \equiv 1 \ \text{ and } \begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} = \sum_{j=0}^{2N+1} N_{j,4}(t) \cdot \mathbf{b}_j \tag{20}$$

with coefficients $\mathbf{b}_0{=}\mathbf{b}_0{}^{(0)}$, $\mathbf{b}_{2i+1}{=}\mathbf{b}_1{}^{(i)}$, $\mathbf{b}_{2i+2}{=}\mathbf{b}_2{}^{(i)}$, $(i{=}0, \ldots, N{-}1)$ and $\mathbf{b}_{2N+1}{=}\mathbf{b}_3{}^{(N-1)}$. Analogously, the B-spline representation of the Euler vector is given by

$$\widetilde{\mathbf{d}}(t) = \sum_{j=0}^{2N+1} N_{j,4}(t) \cdot \mathbf{f}_j \tag{21}$$

with coefficients $\widetilde{\mathbf{f}}_0{=}\widetilde{\mathbf{f}}_0^{(0)}$, $\widetilde{\mathbf{f}}_{2i+1}{=}\widetilde{\mathbf{f}}_1^{(i)}$, $\widetilde{\mathbf{f}}_{2i+2}{=}\widetilde{\mathbf{f}}_2^{(i)}$, $(i{=}0, \ldots, N{-}1)$ and $\widetilde{\mathbf{f}}_{2N+1}{=}\widetilde{\mathbf{f}}_3^{(N-1)}$.

If the spline construction is used as an interpreter scheme of the given (e.g. taught) positions, then the third step can be omitted. The third step is useful for the off-line generation of the spline motion or for some post-processing of the trajectories, as it leads to a reduction of the required data volume.

Tracking the spline motion

We have to generate a sequence of positions relating to a certain time cycle $\Delta\tau$ (typically about 16 milliseconds) for tracking the constructed spline motion by the end-effector of the robot. From these positions the robot control computes the angles of the robot joints and performs a fine interpolation. If we use the motion parameter t as the time it would cause an undesired distribution of velocity along the path in general. Therefore, we have to apply an appropriate *reparameterization* $t = t(\tau)$ (which is assumed to be continuously differentiable) relating the motion parameter t to the time τ. The reparameterization function $t(\tau)$ increases monotonically and satisfies $t(0) = t_0$, i.e. the start position Pos_0 corresponds to the time $\tau = 0$. The velocities and accelerations with respect to the real time τ will be marked by an asterisk *.

After the reparameterization, the absolute value $v^*(\tau)$ of the velocity of the TCP is equal to

$$v^*(\tau) = \|\frac{d}{d\tau}\mathbf{m}(t(\tau))\| = \|\mathbf{m}'(t(\tau)\| \cdot \dot{t}(\tau), \qquad (22)$$

where the dot $\dot{}$ indicates the derivative with respect to the time τ. Similarly, the absolute value $\omega^*(\tau)$ of the angular velocity is

$$\omega^*(\tau) = \|\vec{\omega}(t(\tau))\| \cdot \dot{t}(\tau). \qquad (23)$$

The vector $\vec{\omega}(t)$ is defined as in (6).

The reparameterization has to produce an almost trapezoidal velocity distribution: segments with constant speed are joined by segments with linearly varying speed distribution, cf. Figure 5. These velocity profiles are required in many applications such as welding.

The user can either specify the desired velocity of the TCP or the desired angular velocity of the end-effector. For most segments the user should specify the velocity of the TCP as this is the more intuitive measure of the robot speed. Whereas the speed control by the angular velocity *has to be used* for motion segments with a constant TCP, like between Pos_3 and Pos_4 in Figure 2. Analogously, the speed control by the velocity of the TCP *has to be used* for motion segments with a constant rotational part. (Motion segments with both constant TCP and constant rotational part have to be excluded. These segments are recognized by the interpreter and not transferred to the motion planner.) In order to see if the specified speeds are realistic we check one segment in advance. No global check is being applied as this would destroy the local property of the tracking algorithm.

Consider the i-th segment $t_i \leq t(\tau) < t_{i+1}$ of the spline motion (i=0,...,N-1). If its motion speed is governed by the TCP, then the user specifies the desired absolute values $v^*_{i,\mathrm{seg}} > 0$ and $v^*_{i+1,\mathrm{pos}} \geq 0$ of the velocity of the TCP in the interior of the segment and at the segment's end position Pos_{i+1}. Otherwise the user chooses the desired absolute values $\omega^*_{i,\mathrm{seg}} > 0$ and $\omega^*_{i+1,\mathrm{pos}} \geq 0$ of the angular velocity of the end-effector in the interior of the segment and at the segment's end position Pos_{i+1}. Note that $\mathbf{m}'(t_{i+1}) = \vec{\mathbf{0}}$ implies $v^*_{i+1,\mathrm{pos}} = 0$, see (22), whereas $\vec{\omega}\,(t_{i+1}) = \vec{\mathbf{0}}$ implies $\omega^*_{i+1,\mathrm{pos}} = 0$, see (23). If the user does not specify a value for one of the velocities $v^*_{i,\mathrm{seg}}$ and $v^*_{i+1,\mathrm{pos}}$ or $\omega^*_{i,\mathrm{seg}}$ and $\omega^*_{i+1,\mathrm{pos}}$, then the scheme keeps the preceding value.

We denote by $t^{(k)}$ the parameter value after the k-th time cycle, i.e., we set $t^{(k)} = t(k \cdot \Delta\tau)$. In addition to the specified velocities and angular velocities, one has to choose the maximal and the minimal parameter stepsize $\Delta t_{\max}$ and $\Delta t_{\min}$ which are upper and lower bounds of the differences $t^{(k+1)} - t^{(k)}$ between adjacent parameter values.

The user has to specify the maximal absolute values $a^*_{\max}$ and $\alpha^*_{\max}$ of the *tangential* components of the acceleration and of the angular acceleration. We compute the sequence of positions in such a way that the velocities always satisfy $|\frac{d}{d\tau}v^*(\tau)| \leq a^*_{\max}$ and $|\frac{d}{d\tau}\omega^*(\tau)| \leq \alpha^*_{\max}$. The values $a^*_{\max}$ and $\alpha^*_{\max}$ are related to the actuator acceleration limits of the robot by the Jacobian matrix. This matrix, however, depends on the pose of the robot. In the current implementation we use global bounds $a^*_{\max}$ and $\alpha^*_{\max}$ for accelerations. A more sophisticated possibility could be to choose these bounds according to the geometry of the robot's position, and also depending on the curvature of the robot's path.

In order to realize the desired distribution of the velocities we have to determine the positions where the robot motion has to be accelerated or slowed down. Taking only the tangential components of the acceleration of the TCP and of the angular acceleration of the end-effector into account, these positions do only depend on the arc length of the trajectories. Unfortunately, it is generally impossible to compute the exact arc length of a cubic spline curve. We use lower bounds for the arc length as numerical integrations are too expensive for real-time calculations.

We outline the algorithm for one time cycle of the path generation. The parameter value $t^{(k)}$ and the corresponding position of the end-effector is assumed to be known. According to the specified robot speed we have to compute the next parameter value $t^{(k+1)}$ and the resulting position of the end-effector. Let i be the number of the current spline segment ($0 \le i < N$), i.e. $t_i \le t^{(k)} < t_{i+1}$. If in this segment the robot speed is governed by the velocity of the TCP then the next position results from the following computations:

1. Estimating of the current velocity. We estimate the real velocity $v^*_{k-1,\mathrm{real}}$ of the TCP in the previous time cycle:

$$v^*_{k-1,\mathrm{real}} = \frac{1}{\Delta\tau} \cdot \|\mathbf{m}(t^{(k)}) - \mathbf{m}(t^{(k-1)})\|.$$

For the first time cycle ($k = 0$) we set $v^*_{-1,\mathrm{real}}$=0.

2. Choosing the desired velocity. The next desired TCP velocity of the TCP $v^*_{k,\mathrm{spec}}$ is determined based on the user–specified velocities $v^*_{i,\mathrm{seg}}$ and $v^*_{i+1,\mathrm{pos}}$. If the distance of $\mathbf{m}(t^{(k)})$ to the segment end point $\mathbf{r}_{i+1}$ is small enough, then the motion has to be accelerated or slowed down in order to realize the specified velocity $v_{i+1,\mathrm{pos}}$ at Pos$_{i+1}$, cf. Figure 3. More precisely, if the inequalities $v^*_{k-1,\mathrm{real}} > v^*_{i+1,\mathrm{pos}}$

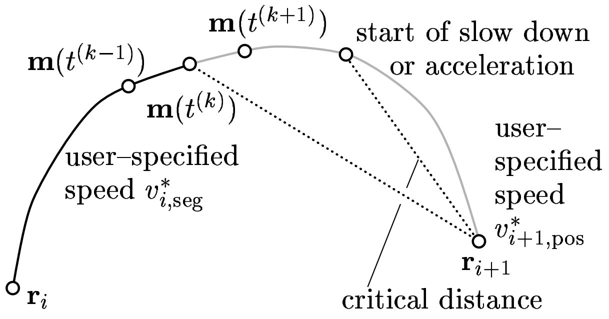

Figure 3: Tracking a spline segment

and

$$\|\mathbf{m}(t^{(k)}) - \mathbf{r}_{i+1}\| \le \frac{1}{2a^*_{\max}} \cdot \Big((v^*_{k-1,\mathrm{real}})^2 - (v^*_{i+1,\mathrm{pos}})^2 \Big)$$

hold, then we have to slow down the motion, hence we set

$$v^*_{k,\mathrm{spec}} = \max\,\{v^*_{k-1,\mathrm{real}} - a^*_{\max} \cdot \Delta\tau\ ,\ \ v^*_{i+1,\mathrm{pos}}\}\,. \quad (24)$$

Otherwise, if the inequalities $v^*_{k-1,\mathrm{real}} < v^*_{i+1,\mathrm{pos}}$ and

$$\|\mathbf{m}(t^{(k)}) - \mathbf{r}_{i+1}\| \le \frac{1}{2a^*_{\max}} \cdot \Big((v^*_{i+1,\mathrm{pos}})^2 - (v^*_{k-1,\mathrm{real}})^2 \Big)$$

hold, we have to accelerate the motion:

$$v^*_{k,\mathrm{spec}} = \min\,\{v^*_{k-1,\mathrm{real}} + a^*_{\max} \cdot \Delta\tau\ ,\ \ v^*_{i+1,\mathrm{pos}}\}\,. \quad (25)$$

These computations are based on the fact that the distance $\|\mathbf{m}(t^{(k)}) - \mathbf{r}_{i+1}\|$ is a lower bound for the arc length of the trajectory of the TCP between the points $\mathbf{m}(t^{(k)})$ and $\mathbf{m}(t_{i+1}) = \mathbf{r}_{i+1}$, see Figure 3. In order to get tighter lower bounds for the arc length one may inscribe a polygon to the spline curve. By using a few polygon points we can approximate the arc length of the trajectory with sufficient accuracy in our implementation.

If the specified velocity $v^*_{k,\mathrm{spec}}$ does neither result from (24) nor from (25), then we try to realize the desired velocity $v^*_{i,\mathrm{seg}}$ which has been specified by the user for the interior of the spline segment. If $v^*_{k-1,\mathrm{real}} > v^*_{i,\mathrm{seg}}$ holds, then we slow down the robot motion,

$$v^*_{k,\mathrm{spec}} = \max\,\{v^*_{k-1,\mathrm{real}} - a^*_{\max}\cdot\Delta\tau\ ,\ \ v^*_{i,\mathrm{seg}}\},$$

otherwise we accelerate the motion,

$$v^*_{k,\mathrm{spec}} = \min\,\{v^*_{k-1,\mathrm{real}} + a^*_{\max}\cdot\Delta\tau\ ,\ \ v^*_{i,\mathrm{seg}}\}.$$

3. Computing of the next position. We compute the next parameter value $t^{(k+1)}$ and the corresponding position of the end-effector. The parameter value $t^{(k+1)}$ should satisfy

$$\|\mathbf{m}(t^{(k+1)}) - \mathbf{m}(t^{(k)})\| = \Delta\tau\cdot v^*_{k,\mathrm{spec}}$$

This non-linear equation for $t^{(k+1)}$ is approximately solved. Let $\Delta t^{(k)} = t^{(k+1)} - t^{(k)}$. We choose the initial guess

$$\Delta t^{(k)}{}_0 = \min\ \{\ \Delta\tau\cdot\frac{v^*_{k,\mathrm{spec}}}{\|\mathbf{m}'(t^{(k)})\|}\ ,\ \Delta t_{\max}\ \}.$$

Afterwards we apply a few steps of the *Regula falsi* (method of false position):

$$\Delta t^{(k)}{}_{j+1} = \min\{\ \frac{\Delta t^{(k)}{}_j\cdot\Delta\tau\cdot v^*_{k,\mathrm{spec}}}{\|\mathbf{m}(t^{(k)}) - \mathbf{m}(t^{(k)} + \Delta t^{(k)}{}_j)\|}\ ,\ \Delta t_{\max}\ \}\qquad (j = 0,1,2).$$

The accuracy should now be sufficient, otherwise one may iterate the Regula falsi. We compute the next parameter value

$$t^{(k+1)} = \min\,\{t^{(k)} + \max\,\{\Delta t^{(k)}{}_3\ ,\ \Delta t_{\min}\}\ ,\ t_N\}$$

We have to increment the current segment number i if $t^{(k+1)} \geq t_{i+1}$ holds.

The tracking of the rational spline motion is completed for $i = N$.

We compute the position $\mathbf{m}(t^{(k)})$ of the TCP and the corresponding rotation matrix $R(t^{(k)})$ from the equations (15) and (17) or from (20) and (21). The evaluation of the (piecewise) polynomials in these equations and their derivatives should be done with the help of the de Casteljau or de Boor algorithms[1].

If the control of the robot speed of the i-th segment is based on the specified angular velocities $\omega^*_{i,\mathrm{seg}}, \omega^*_{i+1,\mathrm{pos}}$ instead of the specified TCP velocities $v^*_{i,\mathrm{seg}}, v^*_{i+1,\mathrm{pos}}$ of the TCP, then the next position of the end-effector results from another algorithm which is completely analogous to the previous one. In this case one has to use the *normalized* Euler vector $(\|\widetilde{\mathbf{d}}(t^{(k)})\|)^{-1}\cdot\widetilde{\mathbf{d}}(t^{(k)})$ and its derivative with respect to t instead of the position $\mathbf{m}(t^{(k)})$ of the TCP and its derivative $\mathbf{m}'(t^{(k)})$, Additionally, the distance $\|\mathbf{m}(t^{(k)}) - \mathbf{r}_{i+1}\|$ between two adjacent TCP positions must be replaced by the angle $2\cdot\arccos\left[(\|\widetilde{\mathbf{d}}(t^{(k)})\|)^{-1}\cdot\widetilde{\mathbf{d}}(t^{(k)})^\top\widetilde{\mathbf{q}}_{i+1}\right]$ of the corresponding rotation $R(t^{(k)})^{-1}\cdot S_{i+1}$. For the sake of brevity we omit the details of the algorithm.

In Figures 4 and 5 we present an example for the construction of a rational spline motion and its tracking by using the above–described method. The seven taught positions have been drawn in Figure 4a, whereas Figure 4b shows the resulting spline motion. The units of the three coordinate axes are 1 mm. The six spline segments (marked by <1>...<6>) are to be tracked with different velocities of the TCP. In Figure 5 we show the desired velocities (in grey) and the resulting velocity after the reparameterization (in black). Both curves are almost identical. An

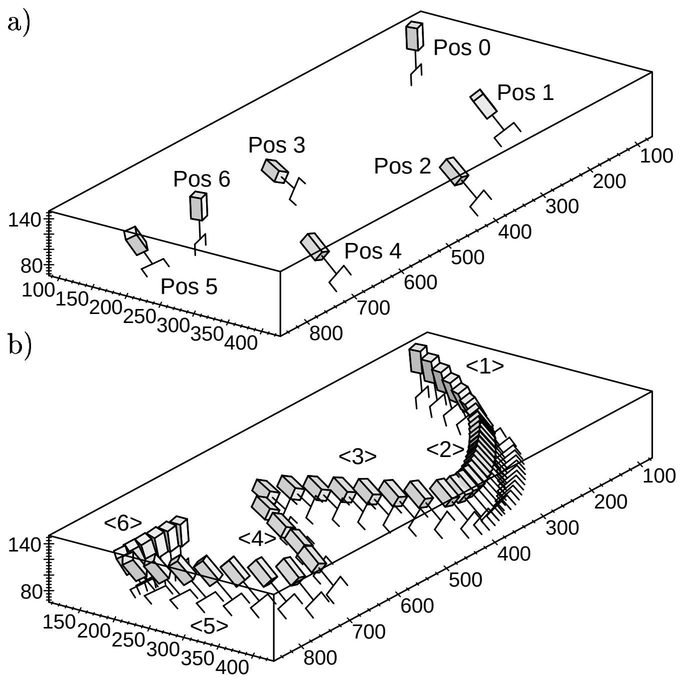

Figure 4: Tracking a rational spline motion: a) the taught positions, b) the spline motion.

almost trapezoidal velocity profile has been achieved between the desired values for the speed as specified by the user.

In principle, the above–described tracking method can be applied to any other Cartesian path description of the robot, provided that points and derivatives can be computed fast enough. As a matter of future research one should derive tighter bounds $a^*_{\max}$ and $\alpha^*_{\max}$ for the tangential accelerations which depend on the robot's position and on the curvature of its path. This would produce a more sophisticated method for dealing with dynamical effects.

An application

The interpolation and tracking algorithms described in the previous sections have been implemented as

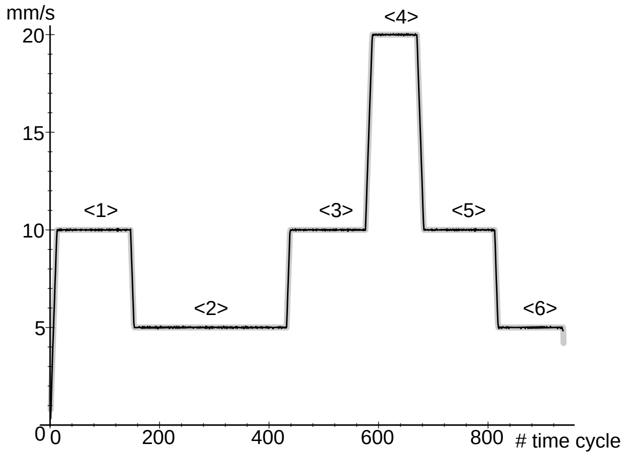

Figure 5: Tracking a rational spline motion: the velocity after the reparameterization.

a part of a new robot controller. Its prototype has been presented at the Hanover Industrial Fair in April 1996. We illustrate it with an example.

This application is taken from a welding task at Odense Steel Shipyard (OSS), a Danish company producing large container vessels. Only ship assemblies consisting of plane segments and straight lines have been welded up to now. The goal in future is to weld also highly complex geometries containing advanced multi–layer welds. Pipe welding is considered as one of the most complex welding tasks.

Figure 6: Intersection of two pipes forming a complex saddle type curve. The welding gun is attached to the robot's flange.

OSS processes up to 45 000 pipes per year. Each pipe consists of two or three different weld types. A lot of them are multi-pass welded with a butt joint in a 3–dimensional curved geometry with full penetration (see Fig. 6). For this task an appropriate robot has been designed: A gantry system (3 translational degrees of freedom) with an attached vertical arm

(3 rotational degrees of freedom) as shown in Figure 7. In addition there is a turn/tilt table in the work

Figure 7: Robot cell at OSS for pipe welding applications.

space in order to turn the pipe into a configuration on which welding from the top should be possible. (This robot configuration is just an example. We have also tested the spline scheme for more complicated robot geometries including 6R manipulators.)

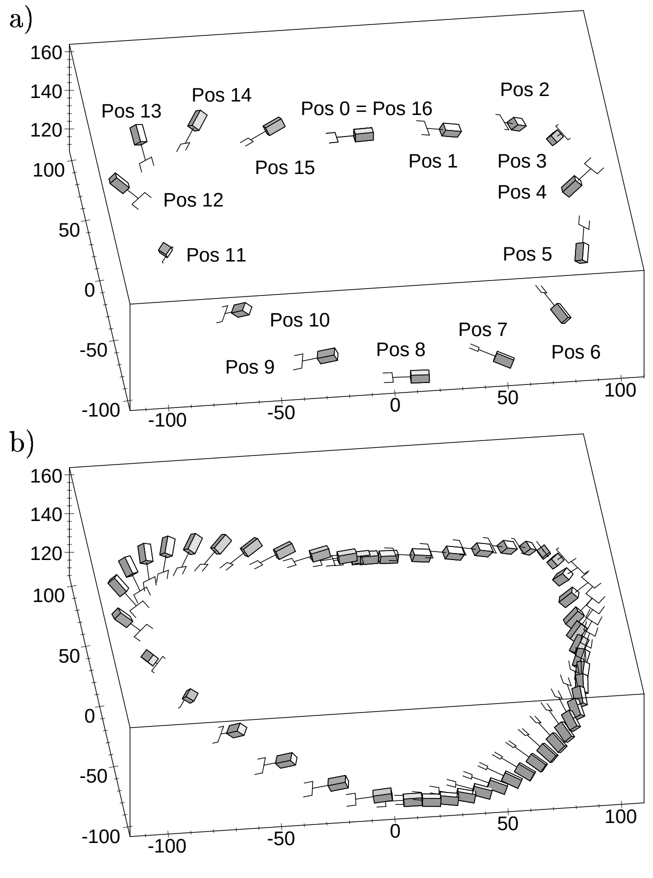

Figure 8: A realistic example. a) The taught positions, b) the interpolating spline motion.

An interpolating spline motion is programmed for this complex curve. The robot program consists of 17 taught positions taken out of a database which consists of nearly all possible welding tasks (Figure 8a). Figure 8b shows the resulting spline motion. We omit the plot of the (almost constant) velocity distribution after the reparameterization as there is virtually no difference between the real and the desired speeds.

In this application, using spline interpolation instead of linear or circular interpolation reduces the number of required taught positions to approximately 30%. Moreover, the interpolating motion is much smoother and the resulting movement of the robot can be performed with higher speed. Therefore, using the spline interpolation leads to a reduction of costs and working time required for the robot programming, and moreover to an increase of productivity.

Concluding remarks

The work described in this paper intends to be a first step towards the use of Computer Aided Design methods in order to solve problems which do arise from Robotics and Kinematics applications. The results of user tests with a prototype implementation look promising. These tests have shown several advantages compared to the traditional method:

- ease of programming (less teach points are necessary),
- higher speed since design of C^1 motion,
- less mechanical stress since design of C^1 motion,
- faster optimization of robot programs.

Future points of interest include

- the direct integration of CAD data into the process of robot programming. This can be done in two different ways. One simple approach is to use the

CAD data for generating positions for the "teach–in" method. As an advantage of this approach the user can easily modify the resulting robot motion. However, the more direct approach would be to use the curve description of the CAD data directly for the robot control. Both approaches require additional information about the desired orientation of the end–effector which is generally not obvious from the current CAD data interfaces. CAR (Computer Aided Robotics) tools like IGRIB or ROBCAD should offer interfaces for incorporating rational motions into their systems in the future. Therefore more sophisticated calibration techniques need to be developed to enhance the model fidelity of robotic systems.

- the optimization of spline motions taking into account robot dynamics (time or energy optimal) Known methods (see for example the textbook by Pfeiffer and Reithmeier[10]), in general used for circular and linear motions can be applied for spline motions as well. Due to the high computational complexity this optimization needs to be calculated off–line (before the robot motion starts).

The authors are not aware of any other commercial system applying the described geometric methods from Computer Aided Design in the field of robotics and believe that this approach will open new possibilities in the future.

Acknowledgment

The authors would like to thank the referees for their comments which have helped to improve the paper.

References

1. Hoschek, J., and Lasser, D., *Fundamentals of Computer Aided Geometric Design*, AK Peters, Wellesley MA, 1993.
2. Craig, J. J., *Introduction to Robotics*, Addison–Wesley, Reading MA, 1986.
3. Fu, K. S., Gonzalez, R. C., and Lee, C. S. G., *Robotics,* McGraw–Hill, New York, 1987.
4. Wang, F. C., and Yang, D. C. H., 'Nearly arc–length parameterized quintic spline interpolation for precision machining,' *Comp. Aided Design* 25 (1993), 281–288.
5. Farouki, R. T., and Shah, S., 'Real–time CNC interpolators for Pythagorean–hodograph curves,' *Comp. Aided Geom. Design* 13 (1996), 583–600.
6. Röschel, O., 'Motion Design — A Survey', *Computer–Aided Design*, this issue.
7. Bottema, O., and Roth, B., *Theoretical Kinematics*, North-Holland, Amsterdam 1979.
8. Johnstone, J., and Williams, J., 'Rational Control of Orientation for Animation,' *Proceedings of Graphics Interface '95*, 179–186.
9. Jüttler, B., and Wagner, M., 'Computer Aided Design with Spatial Rational B–spline Motions,' *ASME Journal of Mechanical Design* Vol. 118 (1996), 193–201.
10. Pfeiffer, F. and Reithmeier E., *Roboterdynamik*, Teubner, Stuttgart.