Robotica (2003) volume 21, pp. 605–614. © 2003 Cambridge University Press
DOI: 10.1017/S026357470300523X Printed in the United Kingdom

Data reduction for manipulator path planning

Werner Korb* and Inge Troch†

(Received in Final Form: May 3, 2003)

SUMMARY

Today, in robot applications continuous paths often result from CAD or other planning tools. We present here an approach to the question that has been discussed for a long time, i.e. how to approximate a given path by a second one in such a way that the latter lies in a tube of given radius ε around the first. The approximation should be a (normally cubic) spline with a small number of breakpoints. The strategy is based on algorithms used in "computer aided geometric design" and is applied to examples from industrial and surgical robotics.

KEYWORDS: Path planning; Data reduction; B-spline; Mathematical optimization; Spline interpolation; Kinematics; Surgical robot.

1. INTRODUCTION

Today, robots are used for a large variety of tasks. Often a robot of a given structure has to perform a complicated task for rather a short time, followed by a new and equally complicated one, etc. As a result, programming of such changing tasks often cannot be performed with the tools provided by the supplier of the robot, and therefore tools for pre-planning must be used. For the definition of the path in Cartesian space, it is convenient to use CAD programs.[1] These tools produce quasi-continuous paths with a huge number of tabular entries. This paper shows how to reduce that number. With this knot-deletiorn procedure, the resulting path must stay in a given ε-tube around the original path.

The performaning of a concrete task is based on reference signals in **joint space** for the controllers of the various links. Therefore, a description is needed of where relative joint angles and/or joint displacements are given as functions of time. Normally, the geometry of the path to be followed is described in **Cartesian** (work) **space** (determined e.g. by CAD programs). Moreover, not only position control may be required, but also control of velocity or acceleration. The last two are not our concern; we concentrate on position control. In any case, the nonlinear geometric relations between joint space and Cartesian work space (*forward and inverse kinematics*) have to be taken into account. Therefore it is suitable and frequently used to model the robot according to Denavit-Hartenberg notation.[2,3]

Normally, the reference signals (in joint space) cannot be described by simple mathematical formulas that allow for quick evaluation during the motion. Further, the geometry of a path in task space often results from CAD tools, and hence, also in the configuration space the path is given in tabular form. Consequently, for a robot with d DOFs, control is frequently based on tables in which each entry is a vector of dimension $d+1$ consisting of a time instant and the corresponding values of joint angles or displacements. For that reason, suitable interpolation between consecutive entries must take place during the robot's movement. This results in the additional requirement that the movement in Cartesian space must be close enough to thc prescribed path. As discussed below, various interpolation methods are possible. Early investigations for interpolation methods in joint space were suggested by Luh and Lin.[4] It is also possible to plan a path in the Cartesian work space, as suggested by Paul[5] or Taylor,[6] to mention only a few.

When the robot tracks the required path, the interpolation and kinematic transformation between the entries of the underlying table is performed repeatedly. Moreover, robot-control systems can be divided into subsystems that are located on different computer systems connected by a local-area network (e.g. based on ethernet), e.g. parts of the control system can be located directly on a robot and not in a control computer to achieve real-time requirements.[7] Reflections such as these result in the necessity of providing tables with a reasonable number of inputs to limit the data transferred via the communication lines.

The interaction between kinematic transformation, interpolation and data reduction is shown in Fig. 1. Here, data reduction means the approximation of a given path by a second one in such a way that the latter lies in a tube of a given radius ε around the original one. The second path is quasi-continuous and has a minimum number of table entries. This problem has been discussed for a long time. The approach given here is based on a "computer aided geometric design". It can be performed in a Cartesian space or in a joint space, depending on the underlying application. The accuracy measure ε has to be chosen correctly, i.e. for path planning in a joint space, ε has to be chosen such that the new Cartesian path will be close enough to the original one. Hence, the choice of ε depends on the concrete task; more details are given below.

The above discussion is also true for the innovative field of surgical robotics. Again preplanning is performed before the surgical intervention in the operating theatre starts. The planning tools convert diagnostic data (CT and MR images)

* (corresponding author). University of Heidelberg (Dept. for Maxillofacial Surgery), Im Neuenheimer Feld 400, D-69120 Heidelberg (Germany). Main parts of this work were developed when Werner Korb worked with the "Institüt für Analysis und Technische Mathematik, Vienna University of Technology" (Austria). E-mail: werner_korb@med.uni-heidelberg.de
† Univ.-Prof. Dr. Inge Troch; Vienna University of Technology (Institüt für Analysis und Technische Mathematik); Wiedner Haupstr. 8–10/114; A-1040 Vienna (Austria).
E-mail: inge.troch@tuwien.ac.at

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

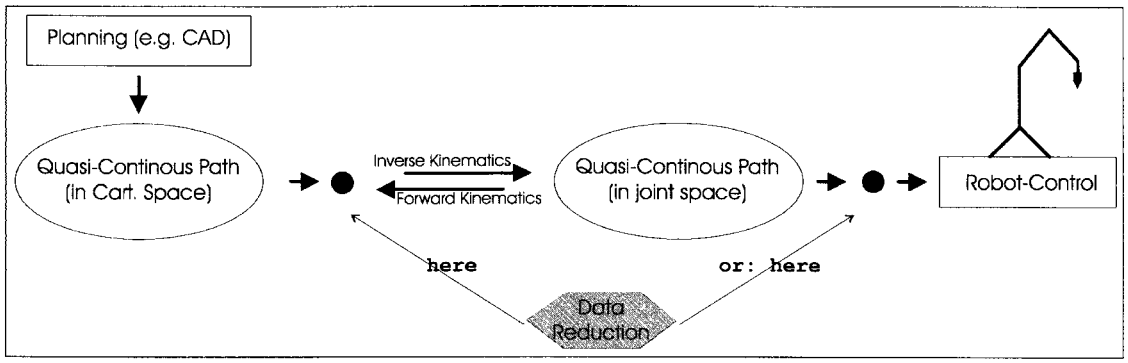


Fig. 1. Schematic drawing of path-planning procedure with knot deletion.

to graphical models, and based on these models, a plan for the surgical operation is developed by the surgeon. One example is the surgical robot *RobaCKa*, which is used in maxillo-facial surgery.[8] Here a plan (milling trajectory) is transfered to the robot controller as a quasi-continuous path with a large number of entries. To obtain a smooth path on the patients skull it is important to find a linear (or cubic) spline with a small number of knots. Further, for safety reasons, the robot control is supported by a vision system, a torque-moment sensor and an overload-protection unit. These systems are all connected and supervised by a Sensor-PC with RTLinux (real-time operating system).[9] The milling procedure can be interrupted at any time for surgical or technical reasons. The Sensor-PC is the supervising unit and is therefore responsible for the execution of the interruption and the possible continuation of the milling procedure. Therefore it transfers knot by knot to the robot-control system. This takes per knot approximately 1/10 sec. For this reason it is important to find a small subset of knots in order to limit the transfer time.

The most important feature of the data minimization algorithm for surgical robotics is the smoothing of the movement by the determination of a cubic spline. The movement strategy of *RobaCKa* is based on a point-to-point strategy from a trajectory point to a trajectory point. This leads to a jerky movement for a large number of knots.

Other path planning methods for surgical robots are known in literature.[10] But they are based on simulation only, (with software tools like, e.g. ROBCAD) and not on the determination of optimal paths with a minimal number of knots.

One early work in that direction of path planning was done by Taylor.[6] He was concerned with the question that the path – resulting from interpolating only a few points in joint space – should be close enough to a straight line in the Cartesian work space. He developed a heuristic procedure to minimize the number of points needed to execute such straight line movements in the Cartesian space by the linear interpolation in joint space. He started with a table in the Cartesian space with entries F_i. For the motion segment (i.e. for two consecutive entries in this table) $F_0 \rightarrow F_1$, the corresponding configurations p_0 and p_1 are computed together with the midpoint $p_m=(p_0+p_1)/2$. Then, the corresponding value F_m in the Cartesian space and the coordinates of the Cartesian path midpoint F_x on the straight line are determined. Then, one tests whether or not a suitable norm of the deviation $(F_x - F_m)$ is less than a given accuracy bound. If this is the case, then nothing has to be done. Otherwise, the corresponding configuration value r_x must be computed and the above steps must be repeated recursively for the motion segments $F_0 \rightarrow F_x$ and $F_x \rightarrow F_1$. This heuristic procedure produces a good, though not necessarily minimal, set of intermediate points.

Our approach is the other way round. Not inserting breakpoints, but deleting superfluous ones from the original data is the goal. Our approach can be applied in any space (joint or Cartesian space). It is only necessary that the new path is sufficiently smooth and lies in an ε-tube around the desired path (and that ε be chosen in the suitable space or at least be chosen appropriately). Our strategy is based on a knot-deletion algorithm used in "computer aided geometric design" (CAGD) and developed by Lyche and Mørken.[11,12] Our goal was developing and testing a procedure for the derivation of a minimal set of breakpoints that can be applied fairly easily for various robotic applications. Consequently, we are not interested in the construction of a path as a function of time through given points and accounting for constraints on, e.g. velocities and accelerations,[5,13–17] but in the derivation of these breakpoints.

This paper is organized as follows: Following this introduction, we give a mathematical formulation of the problem. Then we introduce B-splines and discuss the properties needed for the algorithm (Section 3). In the following sections we discuss the iterative data-reduction strategy and demonstrate that the proposed method can be used successfully in robotic applications in the fields of industry and surgery. Finally, we summarize the results of this work and indicate topics for future research.

In the following we will no longer distinguish between Cartesian and joint spaces because the developed minimization algorithm can be applied in both situations, as shown above (see also Fig. 1).

2. MATHEMATICAL FORMULATION OF THE DATA-REDUCTION PROBLEM

As mentioned above, the prescribed path can be described mathematically in various ways. From the above, the assumption is reasonable that the path of a robot is given as a piecewise linear function with a very large number of nodes. On the other hand, the reference trajectory for the

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

controllers must be reasonably smooth. Normally a continuity of relative joint velocities and joint accelerations is required, sometimes the movement has to be jerk free, i.e. third derivatives of relative joint positions also have to be continuous. Therefore, an appropriate method of reducing the number of breakpoints depends also on the method that is used by the joint controllers for the interpolation between consecutive entries of the table.

This interpolation can be a linear one, as in the early days of robotics, which is still in use for simple tasks where only relative low accuracy is required. Other possibilities[3,18–20] are, e.g. the use of circular, hyperbolic, parabolic or polynomial curves in joint space with transitions between segments with constant velocities according to Paul[21] and, last but not least, spline interpolation. Splines offer advantages, such as the possibility of prescribing the degree of smoothness, the absence of large oscillations between consecutive breakpoints and the availablity of well-tested algorithms with good numerical properties. Our approach is not restricted to the linear interpolation for the reference path or to cubic splines for the approximation. The method can be applied to any type of path such as splines of at least fourth order, which can be used to achieve a smooth and jerk free movement. Cubic splines are favorites in many robotic applications because of second order continuity, i.e. continuity of position, speed and acceleration. Spline interpolation was also suggested in reference [4]. In order to reduce the approximation error, the authors used m interpolation knots to spline $(n-1)$ cubic polynomials together, where $m > n$. Least-squares methods are applied to find the knots, but no idea is given of how to choose the number m of knots.

Besides the choice of the interpolation method, the concrete choice of the accuracy parameter ε is important. First of all, ε depends on the required accuracy in the Cartesian space. If our algorithm is performed in the joint space, then a second parameter $\hat{\varepsilon}$ has to be calculated from the geometric relations between the two coordinate systems in such a way that the accuracy requirements resulting from ε in work space are fulfilled. Due to the many different types of robotic arms, it is not possible to give a general rule for this. However, the appropriate choice will be clear by studying the examples at the end of the paper.

Splines are determined by a set of n given points, $\vec{x}_1, \ldots, \vec{x}_n$. The time needed for their calculation depends on the number n of given points $\vec{x}_k$ $(k=1, \ldots, n)$. Moreover, these calculations have to be performed every time one or more parameters of the desired path are changed; this again emphasizes that breakpoint reduction is necessary.

Therefore, we end up with the following:

Problem formulation: A path is given, described by a piecewise linear function $f=f(x)$ for $x \in [a, b]$, that interpolates a large number n of given breakpoints $x_k \in (a, b)$. A spline $g=g(x)$ of a given type and with $m << n$ breakpoints has to be determined such that $\|f(x)-g(x)\| < \varepsilon$ for all $x \in [a, b]$, where ε denotes a given accuracy bound and $\|\cdot\|$ a suitably chosen norm.

The breakpoints of the resulting function g are the same as those of f. Of course, a more general approach could be to search for the new knots in the ε-tube around f. This would not put any restriction on the original path. On the other hand, this would result in an optimization problem with a huge number of parameters. Such problems are known to be theoretically complex and to require long computation time (if at all solvable).

Splines can be created in various ways.[22,24] B-splines (short for basis-spline) use the fact that splines of a certain order build a vector space. The theory of B-splines is concerned with the choice of a special basis. This theory is fundamental to our strategy. The basis is constructed such that: (i) it depends only on the breakpoints but not on the function to be interpolated, (ii) only a few basis functions (e.g. at most 4 in the case of cubic splines) have values different from zero in a breakpoint, and (iii) the algorithms based on B-spline theory have good numerical properties.

3. DEFINITION AND PROPERTIES OF B-SPLINES

We review those results from the theory of B-splines, which are important for our reduction problem; for details the reader is referred to de Boor[22] or Nürnberger.[25]

The approximation of a function $f(x)$ or interpolation of discrete data on an interval $[a, b]$ by a single polynomial $p(x)$ often leads to large errors and also to convergence problems.[22,23] To overcome these difficulties, the interval $[a, b]$ is divided into m subintervals $[\xi_i, \xi_{i+1}]$ with $a=\xi_1 < \ldots < \xi_m < \xi_{m+1}=b$. On each subinterval $[\xi_i, \xi_{i+1}]$, a polynomial $p_i(x)$ of order k is used for interpolation of $f(x)$ for $i=1, \ldots, m$. Order denotes the maximum number of coefficients $\neq 0$. Further certain continuity conditions at the interior points $\xi_2, \ldots, \xi_m$ must be met. This leads to:

Definition 1 (Spline function)
Let breakpoints $\xi_1 < \ldots < \xi_{m+1}$, a vector $\nu=(\nu_2, \ldots, \nu_m)$ with $0 \le \nu_i < k$ describing the continuity requirements and polynomials $p_i(x)$ of order k be given. (For the sake of compatability of indices, the first element of ν has index 2.) Then the ***spline function*** *$s(x)$ is defined as*

$$s(x) := p_i(x) \text{ on } [\xi_i, \xi_{i+1}] \tag{1}$$

for $i=1, \ldots, m$ with **interior continuity conditions**

$$D^j s(\xi_r^+) - D^j s(\xi_r^-) = 0 \text{ for } 0 \le j \le \nu_r - 1 \tag{2}$$

with $r=2, \ldots, m$ and $D^j s(a^+)$ and $D^j s(a^-)$ the right (left) derivative of $s(.)$ of order j at a. This defines a ***spline space*** *$\mathbb{P}_{k\xi,\nu} := \{s(x) | s(x)$ holds (1) and (2)$\}$.*

Example: The cubic splines result from choosing order $k=4$ and by putting $\nu_i=3$ $(i=2, \ldots, m)$. Piecewise linear functions result for $k=2$ and $\nu_i=1$ $(i=2, \ldots, m)$.

3.1 Definition of B-Splines

The aim of our investigations is the reduction of the number $m+1$ of breakpoints ξ_k. For that purpose, a basis has to be chosen appropriately. A well-known basis with good numerical properties is called **B-splines**. Its definition is based on a knot sequence t. We do not use "vector" and "sequence" in strictly mathematical terms. Both can have a finite or infinite "number of entries". But we require that in a sequence t we have $t_i \le t_{i+1}$. There is a simple relation

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

between these knots t_i, the breakpoints ξ_k and the continuity conditions to be met (see below).

Definition 2 *For a given number k and a given sequence of knots* $t=(t_1, \ldots, t_{N+k})$ *with* $t_i<t_{i+k}$ *for* $i=1, \ldots, N$ *the* ***i-th B-spline*** $B_{i,k,t}$ ***of order*** k ***for the knot sequence*** t *is defined as*

$$B_{i,1,t}(x):=\begin{cases}1 & t_i\le x<t_{i+1}\\ 0 & else\end{cases} \tag{3}$$

$$B_{i,r,t}(x):=\frac{x-t_i}{t_{i+r-1}-t_i}B_{i,r-1,t}(x) + \frac{t_{i+r}-x}{t_{i+r}-t_{i+1}}B_{i+1,r-1,t}(x) \tag{4}$$

for $2\le r\le k$.

However, the functions $B_{i,k,t}$ are building a basis of $\mathbb{P}_{k,\xi,\nu}$ only if the sequence of knots t is chosen appropriately as specified by the theorem of Curry-Schoenberg.[22] For given k and given vectors $\xi=(\xi_l, \ldots, \xi_{m+l})$ and $\nu=(\nu_2, \ldots, \nu_m)$ with $\nu_i<k$ for all i, one calculates first from (1)–(2) the dimension N of the spline space $\mathbb{P}_{k,\xi,\nu}$ as

$$N=k\cdot m-\sum_{r=2}^{m}\nu_r \tag{5}$$

Hence, we need N elements of the basis. Therefore, and in view of (3) and (4), we need a knot sequence t with $N+k$ knots t_i. These knots are selected from the given breakpoints ξ_i such that

$$\begin{aligned}&t_1\le\ldots\le t_k\le\xi_1\\ &<t_{k+1}=\ldots=t_{k+(k-\nu_2)}=\xi_2\\ &<t_{2k-\nu_2+1}=\ldots=t_{2k-\nu_2+(k-\nu_3)}=\xi_3\\ &\ldots<t_{N-(k-\nu_m)+1}=\ldots=t_N=\xi_m\\ &<\xi_{m+1}\le t_{N+1}\le\ldots\le t_{N+k}\end{aligned} \tag{6}$$

That means that $\xi_i\in t$ exactly $(k-\nu_i)$ times (for $i=2, \ldots, m$). Formula (6) shows that the knots $t_1, \ldots, t_k$ and $t_{N+1}, \ldots, t_{N+k}$ can be chosen with some freedom. Often the so-called outer knots are chosen to be $t_1=t_2=\ldots=t_k=\xi_1$ and $t_{N+1}=\ldots=t_{N+k}=\xi_{m+1}$, which would fulfill (6). The other knots, which are defined by (6) in a unique way, are called *interior knots*. For classic cubic splines, all interior knots are single ones. Linear interpolation results if $k=4$ and $\nu_i=1$ for all i, i.e. all interior knots are triple ones.

The basis $B_{j,k,t}$ allows any spline in $\mathbb{P}_{k,\xi,\nu}$ to be represented as

$$s(x)=\sum_{j=1}^{N}\alpha_jB_j \tag{7}$$

with $B_j\in\mathbb{P}_{k,\xi,\nu}$ – given as in (3) and (4) – and with $N=\dim\mathbb{P}_{k,\xi,\nu}$. We write B_j or $B_{j,k}$ instead of $B_{j,k,t}$ if k or t is known from the context.

Note: It is to be recognized from the above that the spline space $\mathbb{P}_{k,\xi,\nu}$ can either be characterized by the order k and the two sequences ξ and ν or by the order k and the knot sequence t together with some coefficient vector α (see below how α is determined, e.g. by interpolation). The latter is emphasized by writing $\mathbb{S}_{k,t}=\{\Sigma_{j=1}^{N}\alpha_jB_j|$ for all $\alpha\}$ instead of $\mathbb{P}_{k,\xi,\nu}$. This is a direct consequence of the Curry-Schoenberg theorem, which leads to $\mathbb{S}_{k,t}=\mathbb{P}_{k,\xi,\nu}$, if and only if (5) and (6).

3.2 More on B-Splines

Interpolation: For a given function $g(x)$ and points $x_1, \ldots, x_N$, we can look for the interpolating spline function $s(x)\in\mathbb{S}_{k,t}$ with $g(x_i)=s(x_i)$ for $i=1, \ldots, N$. Here $x_i\in\mathbb{R}$ and $s:\mathbb{R}\to\mathbb{R}$, but it would be also possible to execute this algorithm for vectors and vector-valued functions.

Representation of $s(x)$ in the form $s=\Sigma_{j=1}^{N}\alpha_jB_j$ leads to a linear interpolation scheme (with unknown coefficients α_j):

$$\sum_{j=1}^{N}\alpha_jB_j(x_i)=g(x_i),\ \ i=1, \ldots, N. \tag{8}$$

This system has a unique solution α for every function $g(x)$, if some conditions hold [22], i.e. for cubic splines, the choice $t_4:=x_1$, $t_{i+4}:=x_{1+2}$ for $i=1, \ldots, N-4$, $t_{N+1}:=x_N$ ensures a unique solution, whereas the remaining knots can be chosen freely. The fact that two points (x_2, x_{N-1}) are not needed for the knot sequence is in agreement with the so called "not-a-knot-end-condition".

The coefficient matrix $B_j(x_i)$ in equation (8) has: (i) band structure (for cubic splines at most 4 values are different from zero on each row) and is, (ii) positive definite.[22] These properties ensure stable and quick solution of (8).

Insertion: For a given order k and a given knot sequence $\tau=(\tau_1, \ldots, \tau_{m+k})$ a linear spline space $\mathbb{S}_{k,\tau}$ of order m results. If we define for the same order k a second, larger knot sequence $t=(t_1, \ldots, t_{n+k})$ with $t\supset\tau$, we get – according to reference [22] – a linear space of dimension $n>m$ with $\mathbb{S}_{k,\tau}\subset\mathbb{S}_{k,t}$. Therefore two representations for functions $f\in\mathbb{S}_{k,\tau}\subset\mathbb{S}_{k,t}$ result with

$$f(x)=\sum_{i=1}^{m}\alpha_iB_{i,k,\tau}(x)=\sum_{i=1}^{n}\beta_iB_{i,k,t}(x) \tag{9}$$

There is a linear relation between the two sets of coefficients,

$$\beta=B\alpha \tag{10}$$

where B is the so called **insertion matrix**. It is needed for the deletion algorithm but also in many other applications of CAGD (rendering, intersection algorithms, etc.) – for details see references [26–28].

4. THE DATA-REDUCTION ALGORITHM

In the following, we perform data reduction for (cubic) splines of order $k=4$. A specific path, e.g. for a d-DOF manipulator is given as $\vec{s}=(s_1(x), \ldots, (x))$. Every spline s^i is

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

defined as an interpolating spline for the same set of breakpoints $x_1, \ldots, x_n$, and hence,

$$s^i(x_j) = y_j^i \quad j = 1, \ldots, n. \tag{11}$$

It is a common assumption in robotics as mentioned already in the introduction – that piecewise linear interpolation $f^i(x)$ ($k=4$, but every inner knot is a triple knot) of the points (x_j, y_j^i) yields the desired path. Therefore $\vec{f} = (f^1, \ldots, f^d)$ is to be considered as the reference trajectory, i.e. as the center of the tube in which the cubic spline must lie that has as few knots as possible.

Consequently, the starting point is a set of vectors

$$\vec{X}_j = (x_j; y_j^1, \ldots, y_j^d) \quad j = 1, \ldots, n \tag{12}$$

that define both, a cubic spline $\vec{s} = (s^1(x), \ldots, s^d(x))$ and the piecewise linear function $\vec{f} = (f^1(x), \ldots, f^d(x))$.

We develop a strategy for reducing the number n of vectors $\vec{X}_j$; so that the new cubic spline $\vec{g} = (g_1, \ldots, g^d)$ defined by a reduced vector set and the piecewise linear function $\vec{f}$ defined by the original vector set $\vec{X}_j$ differ by less than a given tolerance: $\|f^i - g^i\| \le \varepsilon^i$ for $i = 1, \ldots, d$. Here, the deviation is measured by a suitably chosen norm $\|.\|$, and the tolerance may be different for each component of the vector $f - g$.

The basic idea is the assignment of a weight to each knot that indicates the significance of this knot.

We start with the computation of a B-spline representation for the reference function $\vec{f}$ and the corresponding spline $\vec{s}$, both defined by (11). That means determination of knot sequences and coefficient vectors. The knot sequence corresponding to $\vec{f}$ is called $\tilde{t}$. Then we need to check whether $\|f^i - s^i\| \le \varepsilon_i$ holds true for all i. If this is the case, we have at least one solution (namely $\vec{s}$) and can start directly the knot-reduction algorithm suggested by Lyche and Mørken.[12] If the accuracy condition is not met, the insertion of knots into the original knot sequence $\tilde{t}$ must be performed before knot deletion (for details see below).

A twice continuously differentiable cubic spline has to be created in terms of B-spline theory. Therefore, we need a knot sequence $u(t)$ that contains only single interior knots. Consequently, $u(t) \subseteq t$ is determined by taking from t all the outer knots but every interior knot only once. The above can be summarized as follows.

4.1 Mathematical Formulation of the Task

A knot vector $\tilde{t} = (\tilde{t}_i)_{i=1}^{m+4}$, which has only triple interior knots, the order $k=4$ and piecewise linear functions $\vec{f} = (f^1, \ldots, f^d)^T \in \mathbb{S}_{4,\tilde{t}}$ are given together with a vector of tolerances $\vec{\varepsilon} = (\varepsilon^1, \ldots, \varepsilon^d)$ and a suitable norm $\|\cdot\|$ on $\mathbb{S}_{4,\tilde{t}}$. Then, the following two tasks have to be solved in the given order:

(a) *Knot insertion:* Determine the smallest possible knot vector $t \supseteq \tilde{t}$, such that

$$\exists g^i \in \mathbb{S}_{4,u(t)} \text{ with } \|f^i - g^i\| \le \varepsilon^i \ \forall i \tag{13}$$

where t has only triple interior knots.

(b) *Knot deletion:* Find a subspace $\mathbb{S}_{4,\tau}$ of $\mathbb{S}_{4,t}$ of smallest possible dimension and determine $\vec{g} = (g^1, \ldots, g^d)^T$, with $g^i \in \mathbb{S}_{4,\tau}$ such that

$$\|f^i - g^i\| \le \varepsilon^i \text{ for } i = 1, \ldots, d. \tag{14}$$

Notice that problem (a) is equivalent to thc existence of at least one cubic spline within the ε-tube around $\vec{f}$. Further, we give priority to an algorithmic solution of (b) that is easy to use and does not need much computational effort. Hence, we accept *sub-optimal results* for our search.

Whether or not the above requirements for the algorithm can be achieved, depends on the choice of an appropriate norm. The so-called **discrete coefficient norms** approximate the usual L^p norms, [12], and are much easier to handle. Some smart weighted vector norm of the coefficients α of the linear representation $f = \Sigma \alpha_i B_{i,k,t}$ is useful and leads to the definition of the (l^p, t) **coefficient norm of** f as

$$\|f\|_{lr,t} = [\Sigma_{i=1}^{m} |\alpha_i|^p (t_{i+k} - t_i)/k]^{1/p}, \quad 1 \le p < \infty$$

$$\|f\|_{l^\infty,t} = \max |\alpha_i|, \qquad p = \infty$$

The last equation can be written in matrix notation:

$$\|f\|_{l^p,t} = \|E_t^{1/p} \alpha\|_p \tag{15}$$

with

$$E_t^{1/p} = \operatorname{diag}\left\{ \left(\frac{t_{1+k} - t_1}{k} \right)^{1/p}, \ldots, \left(\frac{t_{m+k} - t_m}{k} \right)^{1/p} \right\} \tag{16}$$

for $1 \le p < \infty$ and with $E_t^{1/\infty}$ equal to the $m \times m$ – unity matrix.

4.2 A General Strategy

We explain the strategy for $d=1$ and denote functions $\vec{f} = (f^1)$ simply with f, etc. Extension to general d is straightforward. As stated in Section 4.1, we have to solve the two problems: (a) knot insertion, and (b) knot deletion. The algorithm for problem (a) contains two main components, which are named *approximate* and *insert*. Knot deletion comprises *rank*, *remove* and *approximate*. All components are only applied to the interior knots. The outer knots remain unchanged.

All these components are now explained by using the terminology from above. Insert *approximate* and *ranking* are also discussed in more detail below.

- *approximate* – for given $\tau \subseteq t$ and $f \in \mathbb{S}_{k,t}$, a cubic spline $g \in \mathbb{S}_{k,\tau}$ has to be determined that approximates f. Typically, this process will be some kind of least-squares approximation. The approximation is described formally by thc operator $G(t;\tau)$. This operator represents the approximation process for fixed knot sequences τ and t, hence were determined in *approximate* $g = G(t;\tau) f \in \mathbb{S}_{k,\tau}$.
- *insert* – this component inserts some knots into the original knot vector $\tilde{t}$. This insertion is necessary only if $\|G(\tilde{t}; u(\tilde{t}))f - f\| > \varepsilon$.
- In *rank* we decide in which order knots should be removed from t. Therefore, we build a set $\rho = (\rho_1, \rho_2, \ldots, \rho_n)$ that contains the interior knots of t in a new and fixed (!) order. Now $\rho_1 \in t$ is a knot that is removed if at least one knot is removed, ρ_2 is a knot that is only removed if two or more knots are removed, and so on. For ease of notation we define subsets $\rho^i = (\rho_1, \rho_2, \ldots, \rho_i)$ of ρ that contain the first i elements of ρ. We further define ρ^0, which is the null set. The choice of the knots to be

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

removed is performed with the help of weights as explained later in Section 4.4.

- *remove* performs the determination of the maximum number q of knots that can be removed from t without violation of the perturbation tolerance. This is done by an iterative process. In our case, we used a binary-search algorithm, which performed well.[12,29] Then we build the set $\tau := t \setminus \rho^q$ and apply the operator $G(t;\tau)$ on f, as explained in item *approximate*, which results in $g = G(t;\tau)f$.

Note: It may be necessary to execute *approximate* and *insert* several times, because *insert* may insert not enough knots at one time.

After this, we have three possibilities:

(i) There was no deletion and $t = \tau \supseteq \tilde{t}$ holds. Reasons for this can be an inadequate choice of at least one value ε^i (for $d > 1$) or of the algorithm. In this case, abortion of the iteration is necessary.

(ii) At the other extreme, the sequence τ contains only $2k = 8$ knots. Then the algorithm was very successful because it removed all interior knots.

(iii) In general, we will end up with a sequence $\tau \neq t$ which has more than eight elements. Then we can continue the process iteratively and delete knots from τ.

This iterative process for insertion and deletion is performed by the following algorithm. Here $|t|$ denotes the number of elements of t, further the parameters $it_{(1)}$ and $it_{(2)}$ give the iteration depth of the insertion process and of the deletion process, respectively.

01. *initialisation of insert:* $\tilde{t}^0 := \tilde{t}$; $f_0 := f$;
02. for $r = 0, 1, 2, \ldots, it_{(1)}$
    21. $\tau := u(\tilde{t}^r)$
    22. *determine* $g := G(\tilde{t}; \tau)f_r$;
    23. if ($\|f_r - g\| > \varepsilon$) insert knots $\hat{\tau}^*$ else ($t := \tau$ and go to 03.);
03. *initialisation of deletion*: $\tau^0 := t$; $g_0 := g$;
04. for $j = 0, 1, \ldots, it_{(2)}$
    41. if $|\tau_j| = 8$ then stop;
    42. *determine* ρ^j for $i = 0, 1, \ldots$
    43. $\tau^{j+1} = t^q$, with $t^q = \tau^j \setminus \rho^q$ and $q = \max\{i : \|f - G(t; t^i)f\| \leq \varepsilon\}$;
    44. if $|\tau^{j+1}| = |\tau^j|$ then stop;
    45. $g_{j+1} = G(t, t^q)f$

Computational remarks:

- We want to emphasize that rank is based on the current g_j and not on $g(= g_0)$.
- The choice of $it_{(1)}$ and $it_{(2)}$ is rather sophisticated, because these values do not only depend on ε, but also on the original knot sequence and on various details of implementation. Some experiments must be done to find out the best choice. We had good experience with numbers between 5 and 15 for both.

4.3 Inserting Some Knots

We explain now some main aspects of the particular components. Therefore we return to general d, which is the number of DOFs of the manipulator.

For a given knot sequence $\tilde{t}$, one can find a linear B-spline representation for the piecewise linear functions f^i by determining coefficients d so that

$$f^i = \sum_l d^i_j B_{l,4,\tilde{t}} \quad i = 1, \ldots, d.$$

(see Section 3.2). We want to create cubic splines, therefore we must choose a knot sequence $u(\tilde{t})$ and the corresponding approximating functions $Gf^i = \Sigma_l\, c^i_l B_{l,4,u(t)} = \Sigma_l\, \hat{c}^i_l B_{l,4,\tilde{t}}$, for $i = 1, \ldots, d$. If f^i and Gf^i differ by more than a given ε, one has to insert some knots. For measuring we use the discrete norm

$$\|f^i - Gf^i\|_{(l^\infty,\tilde{t})} = \max_j |d^i_j - \hat{c}^i_j|$$

With the additional knowledge that (see reference [22])

$$d^i_j \approx f^i(\tilde{t}^*_j) \text{ and } \hat{c}^i_j \approx Gf^i(\tilde{t}^*_j) \text{ with } \tilde{t}^*_j = \frac{\tilde{t}_{j+1} + \tilde{t}_{j+2} + \tilde{t}_{j+3}}{3}$$

we can give the insertion-rule:

$$\forall i, j : \varepsilon_i < |d^i_j - \hat{c}^i_j| \text{ insert } \tilde{t}^*_j = \frac{\tilde{t}_{j+1} + \tilde{t}_{j+2} + \tilde{t}_{j+3}}{3} \qquad (17)$$

If $\tilde{t}^*_j \in \tilde{t}$, we insert any other knot nearby. Further, $\tilde{t}^*_j$ must be inserted three times because f^i are piecewise linear functions.

If $\forall i,\ j : \varepsilon^i > |d^i_j - \hat{c}^i_j|$ is true, enough knots have been inserted. The resulting knot sequence is called t. The corresponding coefficients are determined by the insertion matrix in (10).

4.4 Ranking the Knots

The knot sequence $t = (t_1, \ldots, t_{m+k})$ and piecewise linear functions $\bar{f} = (f^1, \ldots, f^d)$ with $f^i \in \mathbb{S}_{k,t}$ are given. First, all triple knots are made into single knots, because we are looking for cubic splines. Then, those knots are determined that should be removed in a single step, by assigning weights and determining a ranking of the weighted knots:

(i) *weight:* Every interior knot of t_j is assigned a weight w^i_j, which depends on f^i. This weight is a rough estimation of the significance of the knot t_j. A knot t_j with small w^i_j is better suited for deletion than one with large w^i_j. This is, of course, only true for the specific f^i under consideration. Therefore we get different weights for every t_j and for every f^i, and hence, a *weight-matrix* $W = (w^i_j)$.

(ii) *rank:* This part determines an order of the knots by consideration of $w^1_j, \ldots, w^d_j$ for every t_j. Then the knots are sorted anew and build ρ.

Rank is performed by the following strategy. We need one single set $\rho^q = (\rho_1, \rho_2, \ldots, \rho_q)$, which will be removed, for all d DOFs. Therefore, we re-sort t with respect to the weight-matrix W. We do this by determination of *ranking numbers* ν_j for every interior knot t_j as

$$\nu_j = \max_{i=1,\ldots,d} \nu^i_j$$

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

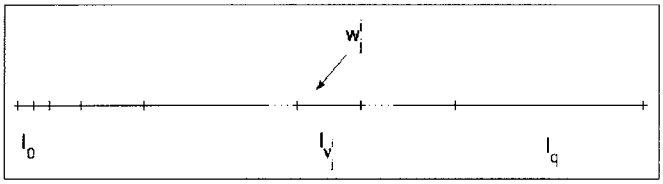


Fig. 2. Intervals I_l^i for *rank*.

where ν_j^i is the number of the interval to which w_j^i belongs (see Fig. 2).

In mathematical terms ν_j^i is determined such that $w_j^i \in I_{\nu_\xi^i}^i$. The intervals are given by:[22,29]

$$I_l^i=[x_l^i, x_{l+1}^i] \quad l=0, \ldots, q-1 \tag{18}$$

with $x_0^i=0$ and

$$x_l^i=s^{i-2}\frac{\varepsilon_i}{\zeta} \tag{19}$$

If $x_q^i<\max_j w_j^i$, a q-th interval $[x_q^i, \max_j w_j^i]$ is added. The data-reduction algorithm always works for $\zeta=1$. In some cases, we saw better convergence for larger ζ. But this is not the topic of this paper and can be investigated in future research.

Then all ν_j are determined, the set ρ, which contains the removable knots t_j, can be computed. The t_j with small ranking numbers ν_j are suitable for removal. Exactly q knots t_j must be chosen. *Weight* is described in more detail in reference [12] and *rank* in reference [11].

4.5 Approximation

We need a good approximation algorithm, because for a knot sequence $t=(t_1, \ldots, t_{m+k})$, in every iteration step approximate is performed about $M=\mathrm{ld}(m-k)$ times (ld means "logarithmus dualis"). For discussion of *approximate* again consideration of $d=1$ is sufficient.

As explained above, in *approximate* the best approximation

$$Gf=\sum_{j=1}^{n} c_j B_{j,k,\tau} \in \mathbb{S}_{k,\tau}$$

for a given function

$$f=\sum_{i=1}^{m} d_j B_{i,k,t} \in \mathbb{S}_{i,k,t}$$

is determined for a given knot sequence $\tau \subseteq t$ and for a given accuracy measure – which is a discrete norm –

$$\|f-Gf\|_{(l^2,t)}=\max_{g\in\mathbb{S}_{k,\tau}} \|f-g\|_{(l^2,t)}. \tag{20}$$

This is a standard linear least-squares approximation. We can write (20) in matrix form and use the insertion matrix B and property (10) for coefficient vectors c and d. But we use a modified matrix notation and minimize

$$\min_{c\in\mathbb{R}^n} \|E_t^{1/2}(BE_\tau^{-1}x-d)\|_2 \tag{21}$$

which has a better numerical condition [12]. $E_t^{1/2}$ is given by (16). From minimizing (21) $c=E_\tau^{1/2}x$ results directly. (Remember that $E^{1/2}$ is a diagonal matrix.)

We used in our implementation the improved Oslo algorithm of reference [28] (in the form of algorithm II) for the computation of the insertion matrix B. To measure the difference of f and g at a specific point x, we measured in $(l\infty, t)$-norm and not in $(1^2, t)$-norm.

5. EXAMPLES AND APPLICATIONS

In this section the algorithm, which has been explained theoretically before, is applied to two different examples from industrial robotics and surgical robotics. In both fields it is important to minimize the data because of two main reasons: (i) the modular architecture needs a repeatedly data transfer from one controller to another, and (ii) the movement is less jerky for robots, which are controlled by a point-to-point movement with stops at each trajectory point.

5.1 Grinding Rear-View Mirrors with a 3-DOF Manipulator

In Fig. 3 a manipulator for grinding rear-view mirrors is given. This robot has 3 joints (rotational, linear and torsional) and holds a mirror (displayed as dark block). For grinding of rear-view mirrors, the first rotational joint is fixed. Hence, a 2-DOF-manipulator results. In Fig. 4 (solid line) typical paths of joints 2 and 3 are displayed (in joint space). The data of these paths yield the piecewise linear function $\vec{f}=(x(t), y(t))$ for which data reduction is applied as discussed before. $x(t)$ is given in mm and $y(t)$ in degrees.

Results of data reduction by using $\varepsilon^1=2$ mm, $\varepsilon^2=2°$ are given in Fig. 4. Attention should be paid especially to the diamonds at the bottom of the figure, which represent the knot sequences (original and reduced). As expected, for the reduced knot sequence most knots remain around the areas of large changes in function values – which correspond to the corners of the mirror. Reduction of data is from $n=248$ to $n=37$. In this case, no data is inserted in the first processing step.

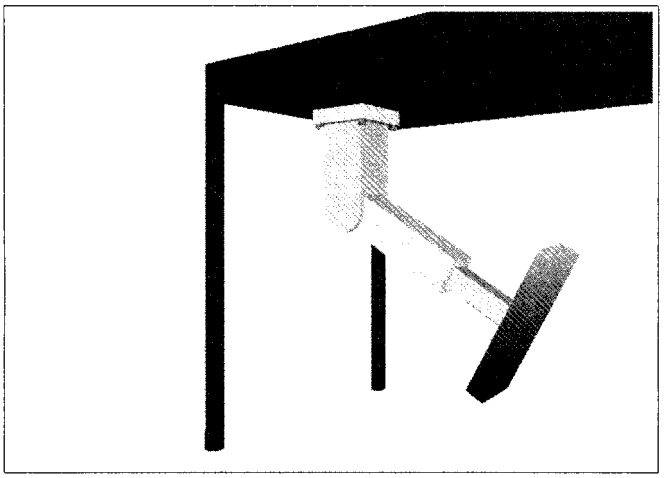


Fig. 3. 3-joint manipulator for grinding rear-view mirrors.

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

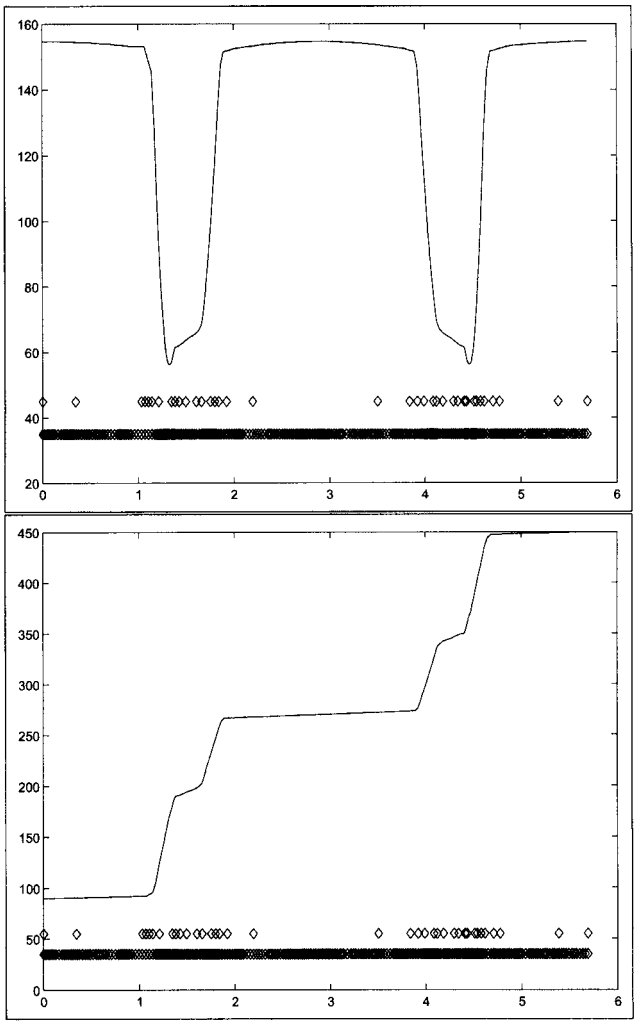


Fig. 4. Visualization of example "Mirror Grinding": Original data and result of applying data reduction. Top: The linear joint. Bottom: The torsion joint. Attention should be paid to the reduction from 248 knots in the original knot sequence to 37 knots in the resulting sequence. Both knot sequences are displayed as diamonds at the bottom of the figures.

Table I. Mirror grinding example: Results of data reduction for different ε (original number of data points $n=248$).

ε	result $\lvert\tau\rvert$	iterations	data inserted	CPU – time
(2, 2)	37	5	0	58.84
(1, 1)	48	5	7	50.33
(0.5, 0.5)	64	6	17	62.95

For different ε the results are given in Table I. Further the CPU time for determination of the algorithm within *Matlab* (TM) on a Pentium III (400 MHZ) – PC is given.

5.2 The 6-DOF Surgical Manipulator ***RobaCKa***

In the second example, a "Staubli RX90" robot (Fig. 5) is used for craniotomy in maxillo-facial surgery. This means milling of a given trajectory on a human skull. Up to now, only tests on phantoms have been performed (Fig. 6). Again knot reduction is applied for the path in the joint space. Because the milling instrument is rotary symmetric, the trajectory is only determined for the first five joints. A

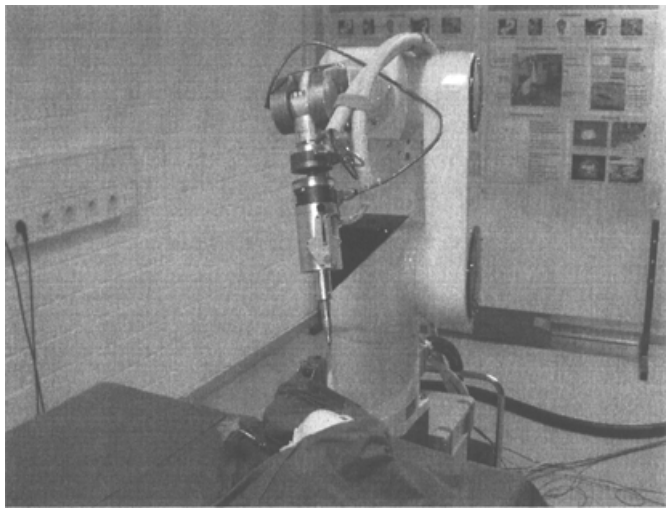


Fig. 5. A Stäubli RX90 robot for use in maxillo-facial surgery.

procedure like that in Fig. 1 is applied: First *surgical planning* is performed. With a planning program, the surgeon displays the human skull as a 3D model and plans the desired milling trajectory. The trajectory is given as a quasi-continuous path in the Cartesian space. Then (the model of the RX90, is given by Denavit-Hartenberg parameters) the determination of *inverse kinematics*, yields a quasi-continuous path in the joint space, to which *data reduction* (as shown above) is applied. For safety reasons, it is important to know the maximal error of the milling trajectory in Cartesian space. This measure is determined as a last step by performing *forwards kinematics* on the reduced data set. If this measure does not give an appropriate value, the reduction procedure has to be repeated with adapted (smaller) values for ε. The original data set is 465 knots, which is a rather detailed and large data set. Therefore, precomputation (*knot insertion*) does not insert any data.

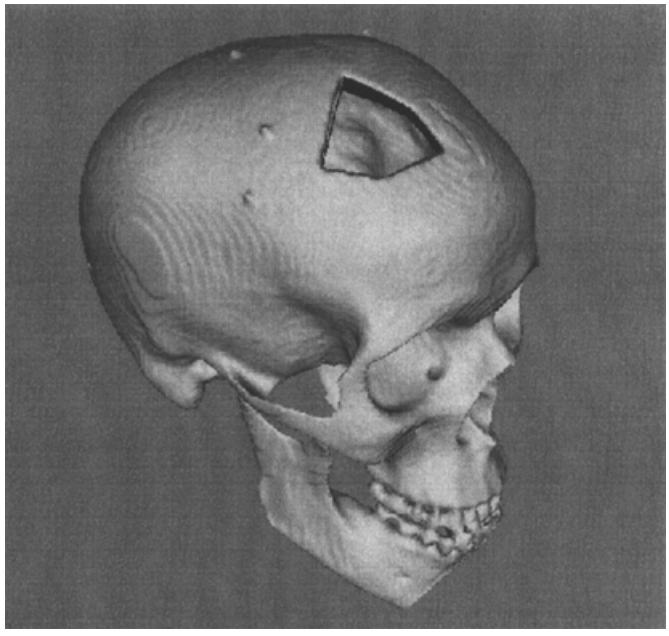


Fig. 6. The objective milling trajectory on the skull. Here a plastic phantom has been cut.

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

Table II. Bone milling example: results of data reduction for different ε (original number of data points $n=465$).

ε (rad)	ε (deg)	result $\lvert\tau\rvert$	iterations	CPU – time	Cart. err. (mm)
0.015	0.859	9	2	147.7	4.8
0.01	0.573	13	2	154.8	2.2
0.005	0.286	24	5	323.6	1.5
0.003	0.172	53	13	939.9	0.7
0.001	0.057	201	11	761.4	0.56

Table III. Data reduction in the Cartesian space for the bone milling example: results of data reduction for different ε (original number of data points $n=265$).

ε (mm)	result $\lvert\tau\rvert$	iterations	CPU – time	Cart. err. (mm)
0.5	33	9	44.1	0.866
0.4	32	13	54.8	0.693
0.3	60	17	80.8	0.520
0.2	139	12	62.2	0.346
0.1	248	10	82.1	0.173

Again we give in Table II a summary of the results of data reduction for different ε. In this case ε is chosen to be the same for all five joints. Again the CPU time is due a Pentium III (400 MHZ). It can be seen that a reasonable reduction is still reached when the (Cartesian) error is not larger than 1 mm.

In this case the results of data reduction in joint space are not very satisfying. This means that there is too large a number of knots in the reduced data set for a comparable large Cartesian error. Furthermore, in surgical robotics driving along the trajectory as fast as possible is not as crucial as in industrial applications. Because of those two reasons, we also perform data reduction in the Cartesian space. Therefore, we have to find a linear spline that lies in the given ε space around the original milling trajectory. As explained above, a solution of this problem can also be determined with the given algorithm. Of course, the movement of the robot would not be smooth (twice differentiable) anymore, and therefore the robot will stop (shortly) at every knot. The results of the data reduction in Cartesian space is given in Table III. ε is equal in the x, y and z directions, and from this the Euclidean error in the Cartesian spacc can be at most $\varepsilon \cdot \sqrt{3}$. This is true, because the linear spline $\vec{g}_i$ with a minimum number of knots lies in the given ε tube around $\vec{f}_i$ for all i ($i=1, 2, 3$), this yields the Euclidean error as $\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_2^2}=\varepsilon\cdot\sqrt{3}$. Here, it can be seen that for the choice $\varepsilon=0.3$, i.e. for a tube with radius of about 0.5 mm, we obtain reduced data sets of reasonable size (about 60 knots).

6. CONCLUSION AND OUTLOOK

It has been demonstrated that data-reduction strategies of CAGD are useful tools in manipulator-path planning. Because of the reduction of knots in quasi-continuous paths, one gains considerable savings of computation time during the robot's movement. These savings are based, e.g. on fewer kinematic computations. An algorithm was applied that deletes knots from splines in "knot-and-coefficient representation". It was demonstrated further, that the algorithm can be applied rather easily for choosing default values for ζ. It is possible to improve the results provided by that standard method. However, to obtain such further improvements, several experiments must be carried out to find appropriate values for certain parameters, such as ζ.

Further reasonable experiments for future research would be:[29] The application of constrained knot deletion.[30] Moreover, algorithms, such as the ones presented in references [21, 31], can be applied to the resulting path in order to obtain a time-optimal movement along this path. Another enhancement is the use of flexible ε tubes. That means that the permitted pertubations ε^i can vary along the time interval $[x_1, x_n]$. This is reasonable in manipulator path planning, e.g. for pick-and-place tasks, because it allows for a varying accuracy according to the specific demands.

Acknowledgements

We thank Martin Damm and Jan Schloen (IRA-TEC *Ingenieursgesellschaft*, Karlsruhe, Germany) for providing data concerning the industrial example. Further D. Engel, S. Hassfeld, J. Mühling and H. Wörn from the collaborative research center SFB 414: "Information Technology in Medical Science. Computer- and Sensor-assisted Surgery" of the DFG (Deutsche Forschungsgemeinschaft) are acknowledged, who supported the authors with the surgical example.

References

1. K.-Y. Young and C.-Y. Huang, "Collision-free path planning and modification based on task requirements". *Mech. Mach. Theory*, **32**(1), 21–37 (1997).
2. R.P. Paul, *Robot Manipulators: Mathematics, Programming, and Control* (MIT Press, Massachusetts, 1981).
3. L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing (Springer, London, 2nd edition, 2000).
4. J.Y.S. Luh and C.S. Lin, "Approximate joint trajectories for control of industrial robots along Cartesian path", *IEEE Trans. on Sys. Man and Cybern.*, **14**(3), 444–450 (1984).
5. R. Paul, "Manipulator Cartesian path control". *IEEE Trans. on Sys, Man and Cybern.*, **9**(11), 702–711 (1979).
6. R.H. Taylor, "Planning and execution of straight line manipulator trajectories." *IBM J. Res. Develop.*, **23**(4), 424–436 (1979).
7. M. Damm, D. Kappey and J. Schloe, "A multi sensor and adaptive real-time control architecture for an autonomous robot". *Proceedings IAS 3* (1993), pp. 439–448.
8. D. Engel, J. Raczkowsky and H. Woern, "Sensor-aided milling with a surgical robot system". **In:** (H.U. Lemke, M.W. Vannier, K. Inamura, A.G. Framan, K. Doi and J.H.C. Reiber, editors) *CARS 2002 – Computer Assisted Radiology and Surgery* (Berlin/Heidelberg, 2002. CARS, Springer), pp. 212–217.
9. D. Engel, J. Raczkowsky and H. Wörn, "A safe system for craniofacial surgery", *IEEE Internat. Conf. on Robotics and Automation*. Seoul, Korea (2001), pp. 2020–2024.
10. L.E. Gonchar, D. Engel, J. Raczkowsky and H. Wörn, "Simulation for path planning and motion control for medical robot". *Workshop on Computer Science and Information Technologies CSIT'2000*, Ufa, Russia (2000), pp. 1–6.
11. T. Lyche and K. Mørken, "Knot removal for parametric b-spline curves and surfaces", *Comput. Aided Geom. Design*, **4**, 217–230 (1987).

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press

12. T. Lyche and K. Mørken, "A data reduction strategy for splines with applications to the approximation of functions and data", *IMA J. Numer. Anal.*, **8**, 185–208 (1988).
13. S. Chand and K.L. Doty, "On-line polynomial trajectories for rohot manipulators". *Int. J. Roh. Res*, **4**(2), 38–48 (1985).
14. Y.-H. Chang, T.-T. Lee and C.-H. Liu, "On-line approximate Cartesian path trajectory planning for robotic manipulator". *IEEE-Trans. on Systems, Man, and Cybernetics, SMC*, **22**(3), 542–547 (1992).
15. C-S. Lin, P.-R. Chang and J.Y.S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots". *IEEE Trans. on Automatic Control,* **AC**, **28**(1), 1066–1073 (1983).
16. S.E. Thompson and R. V. Patel, "Formulation of joint trajectories for industrial robots using b-splines". *IEEE Trans. on Industrial Electronics, IE*, **34**(2), 192–199 (1987).
17. C.-H. Wang and J.-G. Hornig, "Constrained minimum-time path planning for robot manipulators via virtual knots of cubic b-spline functions". *Trans. on Automatic Control*, **35**(5), 573–577 (1990).
18. J.J. Craig, *Introduction to Robotics: Mechanics and Control* (Addison-Wesely, Reading, Massachusetts, 2nd edition, 1989).
19. K. Desoyer, P. Kopacek and I. Troch, *Industrieroboter und Handhabungsperäte* (Oldenburg, München, 1985).
20. K.S. Fu, R.C. Gonzales and C.S.G. Lee, *Robotics: Control, Sensing, Vision and Intelligence* (McGraw-Hill, New York, 1987).
21. I. Troch, "Time-suboptimal quasi-continuous path generation for industrial robots". *Robotica*, **7**, 297-302 (1984).
22. C. de Boor. *A Practical Guide to Splines*, **Vol. 27** of *Applied Mathematical Science* (Springer, 2001. 1st edition 1978, rev. edition 2001).
23. J. Stoer and R. Bulirsch *Introduction to Numerical Analysis* (Springer, Berlin-Heidelberg-New York, 2nd edition, 1993).
24. C. Überhuber, *Computer-Numerik*, **Vol. 1** (Springer, Berlin, Heidelberg, New York, 1995).
25. G. Nürnberger, *Approximation by Spline Functions* (Springer, Berlin, 1989).
26. E. Cohen, T. Lyche and R. Riesenfeld, "Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics". *Computer Graphics and Image Processing*, **14**, 87–111 (1980).
27. R.N. Goldman and T. Lyche, *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces* (Siam, 1993, Chapter 4).
28. T. Lyche and K. Mørken, "Making the Oslo algorithm more efficient". *SIAM J. Numer. Anal.*, **23**, 663–675 (1986).
29. W. Korb, "Datenminimierung und Knoteneduktion bei Splinefunktionen zur Anwendung in der Bahnplanung von Industrierobotern". *Master's thesis* (Vienna University of Technology, 2000).
30. E. Arge, M. Daehlen, T. Lyche and K. Mørken, "Constrained spline approximation of functions and data based on constrained knot removal". **In:** (M.G. Cox and J.C. Maxon, editors) *Algorithms for Approximation II* (Chapman & Hall, 1990), pp. 4–20.
31. I. Troch, "Time-optimal path generation for continuous and quasi-continuous path control of industrial robots". *J. of Intelligent and Robotic Systems*, **2**, 1–28 (1989).

https://doi.org/10.1017/S026357470300523X Published online by Cambridge University Press