29

Smoothing of Piecewise Linear Paths

Michel Waringo and Dominik Henrich
Lehrstuhl Angewandte Informatik III (Robotik und Eingebettete Systeme),
Universität Bayreuth
Germany

1. Introduction

The pointwise traversal of a given path is a popular task in the area of robotics, e.g. in mobile, industrial or surgical robotics. The easiest method to describe paths is by a sequence of linear segments, and for many tasks the precision of a path approximated by linear segments is sufficient. The movements to be accomplished by a mobile robot or the robot's end-effector are described by a sequence of points in space, that have to be traversed by the robot. However, in practice, a pre-computed path unfortunately often consists of more path points than are necessary for sufficiently accurate execution. An excessive number of path points renders the movement jerky if the path points are dispersed around the optimal path, leading to unnecessary mechanical stress of both robot and tool. A second problem is that many path points lying close to one another can lead to high computational cost when traversing the path and can reduce traversal speed.
Paths described by teach-in methods are one example where the path can consist of too many path points. In these methods, the desired movement is recorded while the operator moves the robot's arm, either directly, through a master device or by giving instructions through a control panel. Because of the rather intuitive input of the human operator, the path suffers from deficiencies and frequent unnecessary changes of direction.
The taught-in path can be traversed better after smoothing the path. Another example is voxel-based path planning. Here, only space points with discrete coordinates can be traversed, which may lead to a stair shaped approximation of diagonal paths. Just as with smoothed taught-in paths, smoothed voxel-based paths can be traversed more efficiently, because there are fewer changes of speed and direction, and the total path length is reduced.
The remainder of the text first provides a problem description (Section 2), after which the state of the art is presented (Section 3). Then, the proposed procedure is described in detail (Section 4), and different specializations of the proposed method are shown for points with fixed orientation (Section 5) as well as with variable orientation (Section 6). Finally, experiments are described (Section 7), and open issues and further enhancements are discussed (Section 8).

2. Problem description

The problem of path smoothing can be described as follows. A path $\mathbf{P} := < \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n >$ of n points is given, represented by an ordered list of m-dimensional Cartesian path points

$\mathbf{p}_i := (p_{i1}, p_{i2}, \ldots p_{im})^T$. All path points $\mathbf{p}_i$ have the same dimensionality $m := m_p + m_o$, depending on the degrees of freedom of the robot or the requirements of the task to be performed. The list is sorted in the natural point order, assigning the index 1 to the path's start point and index n to its end point. The parameters $m_p, m_o \in \mathbb{N}$, $m_p, m_o \geq 0$ designate the dimension of the two coordinate types position and orientation of a path point.

The *neighbourhood* of a path point $\mathbf{p}_i$ is defined as the sequence of points in $\mathbf{P}$ between and including the nearest neighbours of that point in the path $\mathbf{P}$ to the left and and right of $\mathbf{p}_i$.

The *deviation* d_i between a smoothed path $\mathbf{P}'$ and the original path $\mathbf{P}$ in the neighbourhood of the path point $\mathbf{p}_i$ can be computed according to various error functions, such as the standard deviation or the area spanning both paths. The deviation must be computed differently depending on which coordinate type, position or orientation, is considered.

The *error function* K represents the criterion used to compute d_i. Its input are the two paths $\mathbf{P}$ and $\mathbf{P}'$ as well as the index i, and its output is the deviation d_i between them. Finally, we need a threshold value d_{lim} indicating the maximum allowed d_i.

If the path points consist of both coordinate types, either position or orientation may, but need not, be used as a constraint in addition to the optimization criterion which is mandatory. If not only an optimization criterion but also a constraint is being used, a second threshold value c_{lim} is needed. In that case, we compute a second deviation c_i for each path point which may not exceed c_{lim}.

Thus, we search for a path $\mathbf{P}'$ whose deviation d_i from $\mathbf{P}$ does not exceed d_{lim} at each individual path point according to K. The number of path points of the path $\mathbf{P}'$ is minimized under the given optimization criteria and optionally a constraint (Figure 1).

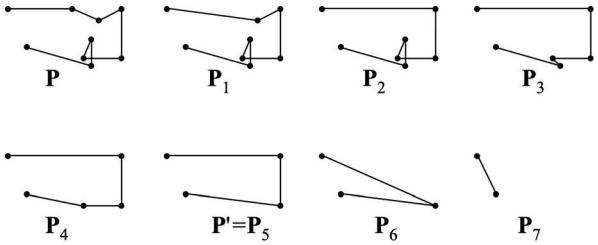

Figure 1. Example of the complete smoothing of a two-dimensional path $\mathbf{P}$ with nine path points. In each step i, the path point whose removal leads to the smallest possible deviation between $\mathbf{P}_i$ and the original path $\mathbf{P}$ is removed, using as criterion the maximum Euclidean distance (see Section 5). A reasonable smoothed path could be e.g. $\mathbf{P}_5$

3. State of the Art

We can distinguish two main categories of problems where a reduced number of path points is required: path planning and path shortening.

In collision-free motion planning, e.g. for mobile robots, the start and the end points are given, and a path connecting them is sought. There may be obstacles and narrow passages like doors or corridors. A good path avoids all obstacles and is short. In a first step, e.g. using a stochastic approach, in a *path planning procedure* (Subsection 3.1), a path of possibly poor quality is created, containing many superfluous segments and being much longer than necessary. It is improved in a second step by a path shortening procedure. There is no need

for any similarity between the original and the collision-free shortened path apart from having the same start and end point.
In other approaches, the improved path must remain similar to the original path. Not only the start and end point, but also the path between them is given. However, the quality of this path may be unsatisfying, the path being jerky or consisting of too many segments. In this *path smoothing* procedure (Subsection 3.2), small deviations from the original path are allowed as long as the number of path segments can be reduced noticably and both paths stay close enough. Figure 2 shows the difference between path shortening and path smoothing.

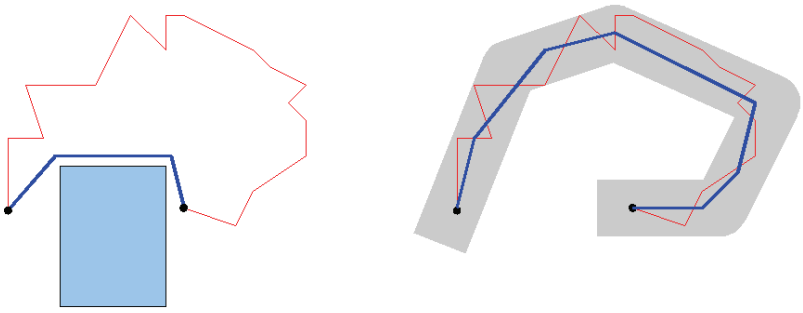

Figure 2. Left: path shortening procedure where obstacles must be avoided, right: path smoothing procedure where a path must stay within a given vicinity of the original path

3.1. Path planning procedures

In collision-free motion planning, planning is usually performed in the configuration space. The problem of finding a path between a start and an end point is PSPACE-hard in the degrees of freedom and in the number of obstacle surface patches. Therefore, most algorithms in this category are stochastic. Two main classes can be distinguished. Probabilistic roadmap (PRM) approaches (Amato, N.M. & Wu, Y., 1996), (Geraerts, R. & Overmars, M. H., 2002) proceed in two steps. First, a collision-free path is constructed as a graph in robot configuration space. In a second step, pairs of promising vertices are chosen and a simple local planner is used to find a better collision-free connection between them. Approaches based on Rapidly-exploring Random Trees (RRTs) (Kuffner, J. J.; LaValle S. M., 2001), (LaValle, S. M., 1998) use a collision-free path tree that is grown incrementally. In each iteration, a random configuration is chosen, and an attempt is made to find a path to it from the nearest RRT vertex.
Other path planning algorithms exist which do not belong to one of these two categories. One example are potential field based methods which can be used for path planning if there are only few obstacles. Another example is the Randomized Path Planner (RPP) (Barraquand, J. & Latombe, J.-C., 1991) where the path is planned according to potential fields and random walks are used to escape from local minima. Another group of path planning algorithms is based on the elastic-band method (Quinlan, S.; Khatib, Q., 1993), where contractive and repulsive forces emanating from obstacles determine the deformation of an original path.

If path segments are to remain piecewise linear, other strategies can be used. Path modifications may be performed by shifting or splitting path segments. Furthermore, path segments can be merged by removing their point of connection (Baginski, B., 1998). A similar procedure is used in (Berchtold, S. & Glavina, B., 1994), where path points are removed based on a heuristic to locally reduce the path length. Another example of collision-free paths for robots can be found in (Urbanczik, C., 2003). Here, path points can be shifted or removed, and segments can be split. After each step the list is re-sorted by pairs of neighbouring segments. Path shortening can be performed using a divide-and-conquer algorithm which removes all path points between the first and the last point in one recursion step if the direct path between them is allowed, and otherwise bisects the path points list (Carpin, S.; Pillonetto G., 2006). However, these approaches are not valid for the application we envisage because they do not guarantee a similarity between the original and the smoothed path.

3.2. Path smoothing procedures

In applications where the shortened path must remain similar to the original path, similar strategies can be used, but optimization criteria are different. The simplest form of path smoothing is the removal of superfluous collinear path points, i.e. path points lying on a straight line between their two immediate neighbours. Here, no deviation from the original path arises, but only a few path points can be removed in general. A reduction in the number of path points can also be achieved by approximating the path by curves of a higher degree consisting of nonlinear path segments (e.g., defined by quadratic or cubic functions) (Hein, B., 2003) or non-uniform rational B-Splines (NURBS) (Aleotti, J.; Caselli, S., 2005).

In (Engel, D., 2003) a smoothing procedure for piecewise linear paths is described that removes path points $\mathbf{p}_j$ not exceeding a given deviation from a path segment $<\mathbf{p}_i, \mathbf{p}_k>$ with $i < j < k$. A disadvantage is that the path point list is treated only once and thus some smoothing steps are not executed which are only possible when the path was already smoothed in a previous step.

A well-known point reduction method is linear regression (Bronstein, I. N.; Semendjajew, K. A; Musiol, G.; Mühlig, H., 2001), but it does not guarantee an upper limit for the deviation. Here, a path defined by scattered points is replaced by a path consisting of one straight-line segment placed as close as possible to the scattered points.

3.3. Conclusions

Although a smoothed path slightly deviates from the original path, it can be better suited for a specific application as long as the deviations are not too big.

A downside of the discussed methods is that they are not able to handle paths with both positions and orientations. They are either restricted to one coordinate type (usually positions) or they work in the configuration space. In that space, there is no differentiation into two coordinate types either. Although algorithms working in the configuration space can smooth paths having position and orientation coordinate components, they need a robot model and a forward kinematics.

The path smoothing method we propose offers some advantages which make it particularly suited for time-critical applications working either in configuration space or work space. Due to the order in which the path points are removed, our method has anytime ability, i.e. it can be aborted prematurely and still returns a valid smoothed path, with the result quality

increasing monotonically over time. The optimization criterion can be easily exchanged (depending on the application), and an upper bound for the deviation between the original path and the smoothed one can be guaranteed. Furthermore, the algorithm is efficient, as both the computation time and storage space are linear in the number of path points. The algorithm is very versatile due to its capability to handle points with both coordinate types.

4. Path smoothing procedure

The path points $\mathbf{p}_i$ are stored in a list ordered by index. Their description (i.e., the Cartesian coordinates) is extended by three components:

- The flag $r_i \in \{true, false\}$ indicates whether the path point has been removed while smoothing.
- The variable $d_i \in \mathbb{R}$ indicates the deviation of the path in the neighbourhood of the path point, which will be explained in detail in Section 5. This variable holds the optimization value used to decide which point has to be removed next so that the path deviation stays as small as possible.
- The variable $c_i \in \mathbb{R}$ stores the deviation of the path in the neighbourhood of the path point according to a second coordinate type. This variable holds the constraint value. The use of c_i is detailed in Section 6. If the path has only one coordinate type, c_i is not used [1].

The path points removed during path smoothing are not deleted from the list, but are instead only marked as removed, since the procedure must be able to access the original path at any time. When smoothing is complete, all path points not marked as removed are copied into a target path point list containing only the path points of the smoothed path.

The algorithm for removing path points works as follows:

(1) For all path points $\mathbf{p}_i$, set $r_i = false$ and $c_i = 0$.

(2) For all path points $\mathbf{p}_k$ not yet removed ($r_k = false$), compute the arising deviation d_k between the smoothed path $\mathbf{P}'$ and the original path $\mathbf{P}$, assuming that $\mathbf{p}_k$ is removed from the path (in addition to the path points removed so far). If a constraint is being used, compute c_k.

(3) Select among all path points with $c_k < c_{\lim}$ the path point p_k with the smallest d_k.

(3a) If the deviation d_k is smaller than the specified value $d_{\lim}$, then mark $\mathbf{p}_k$ as removed ($r_k = true$) and go to (2).

(3b) Otherwise, no further path points can be removed from the path, and the path $\mathbf{P}' := < \mathbf{p}_i \mid r_i = false, i = 1, \ldots, n >$ is returned.

If no constraints are being used, no computations of c_i are performed and c_i stays zero, being without any effect.

The path point whose elimination leads to the smallest deviation from the original path while not violating the constraints is removed during each iteration. In this way, it is ensured that a (locally) maximum number of path points can be removed before the deviation locally exceeds the threshold $d_{\lim}$ and the algorithm terminates.

[1] This is realised by initially setting $c_i = 0$ and not modifying it any more and setting $c_{\lim}$ to an arbitrary value >0.

In order to prevent gradual drifting of the path in each iteration, the current path must not be compared with the path from the previous iteration step, but with the original path.
A certain computational speed improvement can be obtained by using an efficient implementation. Given a path with n path points, the maximum smoothing of the path would require $n - 2$ iterations, as the first and last path point are not removed. For a given iteration step j, the number of local deviation computations is $n - 2 - j$. This belongs to the complexity class O (n^2), since the total number of computation steps is

$$\sum_{i=1}^{n-2}(n-2-i) = \left(\sum_{i=1}^{n-2} n\right) - \left(\sum_{i=1}^{n-2} 2\right) - \left(\sum_{i=1}^{n-2} i\right)$$

$$= n\cdot(n-2) - 2\cdot(n-2) - \frac{n-2}{2}(n-1) = \frac{1}{2}n^2 - \frac{5}{2}n + 3 \qquad (1)$$

The procedure can be accelerated by considering that the path deviation only changes in the proximity of a path point that is removed. Thus only in the first iteration step, the deviation needs to be computed for all path points, and in the further steps only for the path points in the neighbourhood of the last removed path point.
For this purpose, the component d_i of all path points $\mathbf{p}_i$ is required for buffering the corresponding deviation. After a path point has been removed, the deviation can change only for the two neighbouring path points without considering all path points already removed. Therefore, only two instead of $n - 2 - i$ deviations need to be determined per iteration step. This results in a complexity of

$$n + \sum_{j=2}^{n-2} 2 = n + 2(n-3) = 3n - 6 \qquad (2)$$

computation steps, with complexity dropping from O (n^2) to O (n). The same holds true for the computation of the constraint c_i, which, if used, is computed whenever d_i is computed.
In the following, we describe how the interval of path points needed for the computation of the deviation d_i is determined. We are looking for two path points $\mathbf{p}_{min}$ and $\mathbf{p}_{max}$ that border on the interval in question: $I := [\mathbf{p}_{min} ; \mathbf{p}_{max}] = < \mathbf{p}_{min}, \dots, \mathbf{p}_i, \dots, \mathbf{p}_{max} >$.
In the first iteration no path points have been removed yet. Trivially, only three path points need to be regarded: the path point $\mathbf{p}_i$ as well as its neighbours $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$, and the path segment $< \mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1} >$ must be compared with $< \mathbf{p}_{i-1}, \mathbf{p}_{i+1} >$ in order to compute d_i. The manner in which this comparison is performed depends on the error function used and is described in Section 5.
In the subsequent iterations we must consider which path points are removed and must extend the path interval of interest beyond the previously removed path points so that its borders again consist of the next two non-removed path points $\mathbf{p}_{min}$ and $\mathbf{p}_{max}$. Thus $\mathbf{p}_{min}$ and $\mathbf{p}_{max}$ are the direct neighbours of $\mathbf{p}_i$ that have not yet been removed. The deviation d_i is computed by comparing a path segment of the original path $\mathbf{P}_{i,o} = [\mathbf{p}_{min}, \mathbf{p}_{max}] = < \mathbf{p}_{min}, \dots, \mathbf{p}_i, \dots, \mathbf{p}_{max} >$ and the corresponding path segment on the smoothed path $\mathbf{P}_{i,s} = < \mathbf{p}_{min}, \mathbf{p}_{max} >$.
The following table shows exemplarily the deviation computations necessary for smoothing a path with six path points $\mathbf{P} = < \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6 >$ in the order $\mathbf{p}_4$, $\mathbf{p}_3$, $\mathbf{p}_2$ and $\mathbf{p}_5$.

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$	$\mathbf{p}_6$
compute	d_1	d_2	d_3	d_4	d_5	d_6
remove point				r_4=true		
recompute			d_3		d_5	
remove point			r_3=true			
recompute		d_2			d_5	
remove point		r_2=true				
recompute	d_1				d_5	
remove point					r_5=true	
recompute	d_1					d_6

Table 1. Steps performed during the smoothing of a path. In each iteration other than the first one only two deviations need to be determined

Prior to each iteration step, the deviation d_i is known for all remaining path points $\mathbf{p}_i$ not yet removed (thus all path points with r_i = false). Based on this information, the path point whose removal leads to the least deviation from the original path can reliably be removed.
Fig. 3 and Fig. 4 illustrate the two steps marked in gray from Table 1 based on a two-dimensional geometry. For example, the area between a path segment of the smoothed path and the appropriate original path is defined as the error function K. Using this K, the deviations d_i are areas, which are shown in gray.
The path point $\mathbf{p}_4$ is already marked as removed (represented by a white dot in Figure 3 (a)). $\mathbf{p}_3$ is now removed additionally, resulting in the modification of the smoothed path that can be seen in Figure 3 (b) and (c).

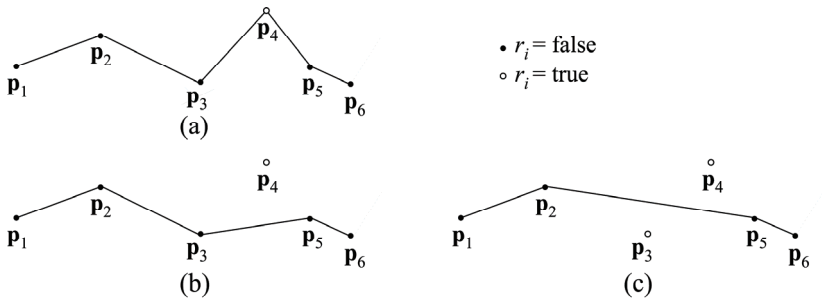

Figure 3. Example for the removal of a path point. Black dots represent still existing path points and white dots represent removed path points. (a) shows the entire path and (b) and (c) the smoothed path before and after removal of $\mathbf{p}_3$, respectively

With the removal of $\mathbf{p}_3$, the deviations d_2 and d_5 occurring upon removal of $\mathbf{p}_2$ and $\mathbf{p}_5$ also change and d_2 and d_5 must therefore be updated. Figure 4 (a) and (b) clarify why the deviation must be recomputed for the path point $\mathbf{p}_2$. Before the removal of $\mathbf{p}_3$, the removal of $\mathbf{p}_2$ only affected the path segment $< \mathbf{p}_1, \mathbf{p}_3 >$. Now, it affects the path segment $< \mathbf{p}_1, \mathbf{p}_5 >$. In the smoothed path, $\mathbf{p}_2$ now has $\mathbf{p}_1$ and $\mathbf{p}_5$ as direct neighbors rather than $\mathbf{p}_1$ and $\mathbf{p}_3$, thus its deviation has changed.
Similarly, in Figure 4 (c) und (d), the deviation for the path point $\mathbf{p}_5$ determined in an earlier iteration is now invalid and must be recomputed. No new calculations need to be performed

for the other path points, since with linear interpolation the removal of $\mathbf{p}_3$ only affects the segments that were direct neighbors of that path point before it was removed.

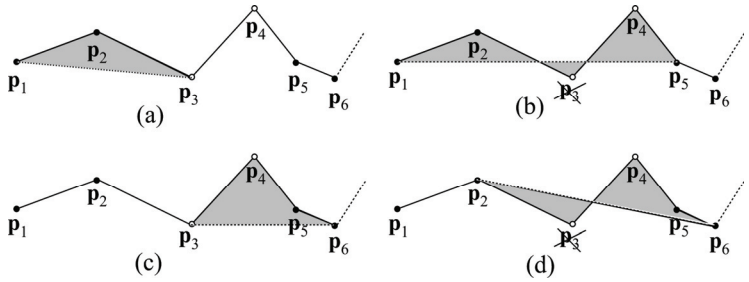

Figure 4. Example of the computational cost reduction. Black dots represent still existing path points and white dots represent removed path points. The deviations are shown as gray areas. In the upper two figures, the deviation d_2 occurring if $\mathbf{p}_2$ is removed is shown, both before removal of $\mathbf{p}_3$ (a) and after its elimination (b). (c) and (e) similarly show the deviation d_5 before and after removal of $\mathbf{p}_3$

In the following we describe how the interval of path points is determined that is needed for the computation of the deviation d_i. We are looking for two path points $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ that border the interval in question: $I = [\, \mathbf{p}_{\min} ; \mathbf{p}_{\max} \,] = \langle \mathbf{p}_{\min}, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_{\max} \rangle$.
In the first iteration no path points have been removed yet. Trivially, only three path points need to be regarded: the path point $\mathbf{p}_i$ as well as its neighbors $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$, and the path segment $\langle \mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1} \rangle$ must be compared with $\langle \mathbf{p}_{i-1}\ \mathbf{p}_{i+1} \rangle$. The manner in which this comparison is performed depends on the error function used and is described in Section 5.
In the subsequent iterations we must consider which path points are removed and we must, as shown in Figure 4, extend the path interval of interest beyond the previously removed path points so that its borders again consist of two non-removed path points. Let i be the index of the path point for which the deviation of the deletion is to be computed and $\mathbf{p}_i$ be the corresponding path point. Let n be the number of path points in the original path $\langle \mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_n \rangle$.
We obtain the following algorithm:

```
min := i – 1
while min > 1 and r_min = true
    min := min – 1

max := i + 1
while max < n and r_max = true
    max := max + 1
```

Thus $\mathbf{p}_{\min}$ und $\mathbf{p}_{\max}$ are the direct neighbors of $\mathbf{p}_i$ that have not yet been removed. The deviation d_i is computed by comparing a path segment of the original path $\mathbf{P}_{i,o} = [\, \mathbf{p}_{\min}, \mathbf{p}_{\max} \,] = \langle \mathbf{p}_{\min}, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_{\max} \rangle$ and the corresponding path segment on the smoothed path $\mathbf{P}_{i,s} = \langle \mathbf{p}_{\min}, \mathbf{p}_{\max} \rangle$.

5. Path deviation functions for a fixed orientation

In this section, we consider only paths with positional coordinates and no orientation and we do not use any constraints (Waringo, M.; Henrich D., 2006). Depending on the application, different error functions K can be used. We investigated three error functions:

- K_1: d_i is the maximum of the Euclidean distances of all path points of the interval $\mathbf{P}_{i,o}$ from the corresponding interval $\mathbf{P}_{i,s}$ in the smoothed path. This criterion can be used in applications where motion is constrained to a safety corridor, e.g. in a master-slave robotic guidance system, car manufacturing, or robotic endoscope holding systems.
- K_2: d_i is the root-mean-square deviation of the distances (i.e. the square root of the mean of the squares of the shortest distances) of all path points of the interval $\mathbf{P}_{i,o}$ from the corresponding interval $\mathbf{P}_{i,s}$ in the smoothed path. This criterion is useful mainly for theoretical analysis.
- K_3: d_i is the area between the smoothed paths segment $\mathbf{P}_{i,s}$ and the corresponding segment $\mathbf{P}_{i,o}$ on the original path. K_3 is the best choice for sweeping applications, e.g. bones milling or cleaning robots.

The first two error functions can be determined quickly and work for path points with any dimensionality. Error function K_3 is useful and intuitively plausible for paths defined in a plane, i.e. two-dimensional paths. However, K_3 can also be used for more dimensions.

The error function K_1 guarantees that the smoothed path never deviates by more than the distance d_{lim} from the original path. One disadvantage involves the computation of each deviation d_i: Only one path point $\mathbf{p}_i \in [\, \mathbf{p}_{\text{min}} ; \mathbf{p}_{\text{max}} \,]$ is used and the distance from the other path points in that interval is neglected. Path points far away from $\mathbf{P}_{i,s}$ are rated strongly, whereas a constant slight deviation of the path across all path segments under consideration leads to a smaller deviation.

This drawback can be avoided by using the error function K_2 as this function uses all path points $\mathbf{p}_i \in [\, \mathbf{p}_{\text{min}} ; \mathbf{p}_{\text{max}} \,]$ for the computation. Additionally, path points far away from $\mathbf{P}_{i,s}$ are considered because the distance to the smoothed path $\mathbf{P}_{i,s}$ is squared. The computation of K_2 is also quite fast.

The computation of K_3 is more costly, however unlike K_1 and K_2 it also considers the distance between path points $\mathbf{p}_i \in [\, \mathbf{p}_{\text{min}} ; \mathbf{p}_{\text{max}} \,]$ on the original path, not just the distances between paths points from the original path and the smoothed path $\mathbf{P}_{i,s}$.

Figure 5 illustrates the three error functions.

The algorithm uses the heuristic of always removing the point yielding the smallest deviation. Although this provides good results in practice, the path obtained is not necessarily globally optimal. Because the algorithm does not look ahead to try to remove more than one path point at a time and does not allow the deviation to exceed the limit d_{lim} in any iteration step, opportunities to shorten the path can be missed. Consider for example Figure 6 (a). When using criterion K_1 and a maximum allowed deviation $d_{\text{lim}} = 0.6 \cdot \| \mathbf{p}_2 \mathbf{p}_3 \|$ the optimal path (b) cannot be obtained. The removal of either $\mathbf{p}_2$ or $\mathbf{p}_3$ temporarily leads to a deviation that is close to $1 \cdot \| \mathbf{p}_2 \mathbf{p}_3 \|$, whereas by removing $\mathbf{p}_2$ and $\mathbf{p}_3$ at the same time, d_{lim} would not be exceeded. The smoothing procedure aborts without being able to remove $\mathbf{p}_2$ or $\mathbf{p}_3$.

Nevertheless, the paths created are valid because the deviation does not exceed the maximum allowed. An advantage of the proposed method is that the algorithm is anytime capable, i.e. it can be aborted prematurely and still delivers a valid result. The quality of the

path increases monotonically until termination. This is an important feature for time-critical applications, such as sensor-based motion planning.

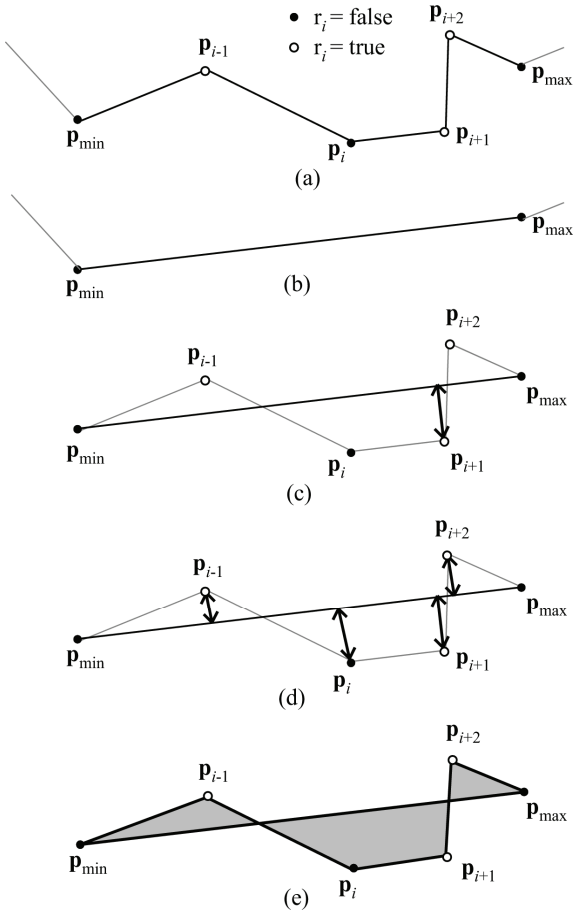

Figure 5. Sketch describing the determination of path deviation d_i. The linear path segments to be compared are the original path (a) and the smoothed path (b). The error functions K_1, K_2, and K_3 are illustrated in (c), (d), and (e)

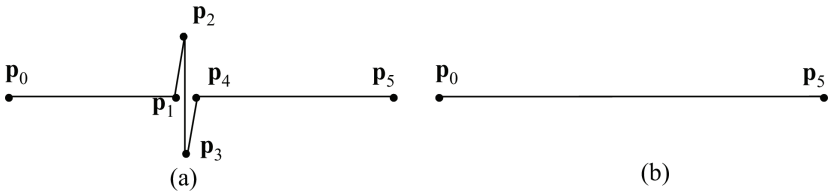

Figure 6. Example for the non-optimality of the proposed algorithm. (a) A path consisting of five points, (b) the optimal path with a maximum allowed deviation of $0.6 \cdot \|\mathbf{p}_2\,\mathbf{p}_3\|$

6. Path deviation functions for variable orientations

In Section 5 paths with arbitrary dimensionality, but just one coordinate type have been treated. However, Cartesian paths whose points contain information on both coordinate types (position as well as orientation) cannot be handled reasonably with this approach, as positions and orientations can not be treated in the same way. For example, a positional value is unique, whereas an orientation is unique in a range of 360°.
Therefore, we choose to compute position and orientation separately and combine positional and orientational deviation in an optimization procedure as optimization criterion and/or constraint, respectively. At that point, both are real numbers which are comparable again.
We use quaternions for representing the orientation of a path point, following the representation in (Maillot, P., 1990). This representation overcomes severe disadvantages of a vector angle representation like e.g. Euler angles.
Eq. (3) shows the representation of an orientation $o_i = \varphi_i$ as a quaternion.

$$\varphi_i = a_i + b_i \cdot j + c_i \cdot k + d_i \cdot l \text{ with } a_i, b_i, c_i, d_i \in \mathbb{R}. \tag{3}$$

Just like the distance between two positions $\mathbf{p}_1$, $\mathbf{p}_2$ can be computed, we can easily obtain the distance between two orientations $\mathbf{o}_1$, $\mathbf{o}_2$. It corresponds to the angle between the orientations (Eq. 5):

$$d_p(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{(p_{2,x} - p_{1,x})^2 + (p_{2,y} - p_{1,y})^2 + (p_{2,z} - p_{1,z})^2} \tag{4}$$

$$d_o(\mathbf{o}_1, \mathbf{o}_2) = \text{acos}\,(a_1 \cdot a_2 + b_1 \cdot b_2 + c_1 \cdot c_2 + d_1 \cdot d_2) \tag{5}$$

From Eq. (5), it is obvious that the distance between two orientations can not exceed the range [–180°, +180°] whereas the distance between two points is unrestricted, Eq. (4).
The criteria K_1 and K_2 are directly applicable for orientations if we replace Eq. (4) by Eq. (5) in the computation. For criterion K_3, we need to obtain the cumulated orientational deviation. We solve that problem by numerically integrating the orientational deviation along the path between two path points. We need to interpolate orientations. Quaternion interpolation can be performed using either LERP or SLERP interpolation (Maillot, P., 1990). Although SLERP interpolation is slightly more computationally expensive, we chose to use SLERP, as LERP interpolation yields only an approximated result. Not interpolating at all but only computing the angle difference between path points $\mathbf{p}_k \in \mathbf{P}_{i,o}$ and the corresponding path points of the smoothed path $\mathbf{P}_{i,s}$ would also only give an approximation.
SLERP interpolation works as follows. Let $p = 0, \ldots 1$ be a control parameter, $\mathbf{q}_1$ and $\mathbf{q}_2$ two orientations given as quaternions and the angle θ between $\mathbf{q}_1$and $\mathbf{q}_2$, as computed in Eq. (6), (7) and (8).
We obtain the control variables

$$r_1 = sin\,((1 - p) \cdot \theta) / sin\,(\theta) \tag{6}$$

$$r_2 = \sin\,(p \cdot \theta) / \sin\,(\theta) \tag{7}$$

and the SLERP-interpolated orientation

$$\mathbf{q}' = r_1 \cdot \mathbf{q}_1 + r_2 \cdot \mathbf{q}_2. \tag{8}$$

By summing the orientational deviations of a smoothed path segment from the corresponding unsmoothed segment, we obtain the orientational deviation using criterion K_3. For this, we do not directly use the unsmoothed path, but the segmentwise projection of the unsmoothed path onto the smoothed path. To illustrate this idea, we show the interpolation of the orientation for the orientational dimensionality $m_o = 1$ and the obtained deviations as well as the sum of the deviation (Figure 7).
For $m_o = 2$ or $m_o = 3$, the procedure works in exactly the same way, as Eq. (5) and Eq. (8) are handling 3D orientations. Orientations of higher dimensionality, although untypical in practical applications, can also be easily used if the path deviation functions are adapted accordingly, just like positions of higher dimensionality are usable.

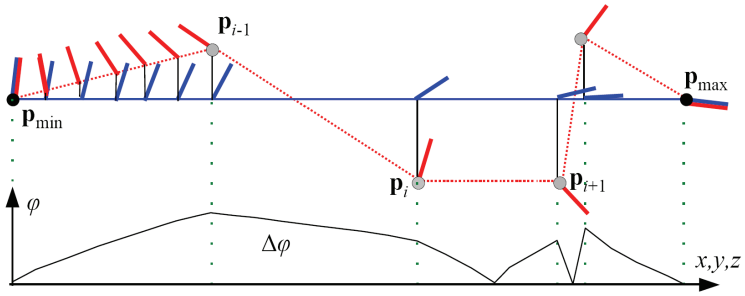

Figure 7. Sketch visualizing the computation of the angle sum $\Delta\varphi$. For simplification, the angle is represented only in 1D. Top: Curve progression of the orientation of a smoothed path [$\mathbf{p}_{\min}$, $\mathbf{p}_{\max}$] (blue) and the original path [$\mathbf{p}_{\min}$, … , $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, $\mathbf{p}_{i+1}$, … , $\mathbf{p}_{\max}$] (red). Bottom: Curve progression of the angle deviation φ between the two paths

In a simple optimization procedure, the computed positional and orientational deviations p_i and o_i are assigned to the optimization and constraint value c_i and d_i. If not both coordinate types are used, all c_i stay zero and the algorithm works without any constraint. On the other hand, the optimization value d_i is indispensable. We obtain five different cases, as shown in Table 2.

	Case				
	(a)	(b)	(c)	(d)	(e)
Optimization value d_i	p_i	o_i	p_i	o_i	$\frac{p_i}{p_{\max}} + \frac{o_i}{o_{\max}}$
Max. optimization deviation $d_{\lim}$	$p_{\lim}$	$o_{\lim}$	$p_{\lim}$	$o_{\lim}$	2
Constraint value c_i	/	/	o_i	p_i	/
Max. constraint deviation $c_{\lim}$	/	/	$o_{\lim}$	$p_{\lim}$	/

Table 2. Enumeration of the five possible cases when combining both coordinate types. The table entries indicate the assignments of the path points positional and orientational deviation p_i and o_i to the algorithm's values c_i and d_i, as well as the assignments of the indicated positional and orientational deviation limit values $p_{\lim}$ and $o_{\lim}$ to the algorithm's values $c_{\lim}$ and $d_{\lim}$

Case (a) is the case seen in Section 5. No orientational information is evaluated, and there is no constraint. The computed positional deviation p_i is used as d_i in the path smoothing procedure. Similarly, in Case (b), we are smoothing orientations without using constraints. The criterion function used to compute d_i is the version adapted to orientations (Eq. (5)), and the deviation d_i used in the optimization procedure is the orientational deviation o_i.
In Case (c), the positional deviation is used as in (a) as optimization value, but in addition we use the orientation as constraint. As long as the limit is not exceeded ($c_i < c_{\text{lim}}$) in any point with the lowest positional deviation, the path is smoothed as in Case (a). Points whose orientational deviation o_i exceeds o_{lim} are not removed, no matter how low p_i is. Case (d) is analogue to Case (c), the roles of position and orientation being exchanged.
In Case (e), we have to optimize the position and orientation simultaneously. Because they are incompatible, we normalize p_i and o_i to p_{lim} resp. o_{lim}, giving an indication on how close both deviations are to the limit and bringing them into the same domain. Now, we can simply sum up both values to obtain d_i. The ratio between p_{lim} and o_{lim} correlates linearly with the relation of the influences of position and orientation. As both $p_i / p_{\max}$ and $o_i / o_{\max}$ are positive numbers and their sum may not exceed 2, both positional and orientational deviations are restricted to twice the maximum indicated.

7. Experiments and Results

In this section, we present and analyze a practical surgical application of our algorithm. First, we compare the results yielded when applying the three error functions described previously. Then, we briefly describe the original paths used in the surgical application and their drawbacks. Finally, we describe the improvement achieved by applying the smoothing algorithm.
In the first rather theoretic experiment, a perturbed linear path consisting of 1000 points, positioned equidistantly on the x axis in the interval [0; 1000] with increasing x coordinate and distributed uniformly on the y axis with y values between –10 and +10, is smoothed using criterion K_1 (maximum deviation). The y coordinate of the first and the last point is 0 (Figure 8).

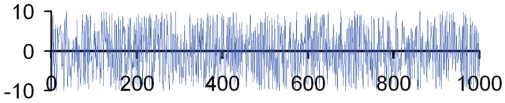

Figure 8. Experimental perturbed path. The units on both axes are millimetres

The experimental results in Figure 9 demonstrate the anytime property of the algorithm. It can be aborted at any time. With a maximum deviation of only 1 mm, reached after 20 ms, already one third of the path points could be removed. The correlation of computation time and number of path points removed is nearly linear. After less than half of the time needed for complete smoothing, half of the path points have been removed.
However, this experiment also shows the sub-optimality of the algorithm. Because the first and the last point have a y coordinate of 0 and the y coordinate of all other points lies in the interval [-10; 10], the path could be reduced to its start and end point with a maximum deviation d_{lim} =10 mm. Yet, in general the algorithm finds that solution only at d_{lim} =20 mm.

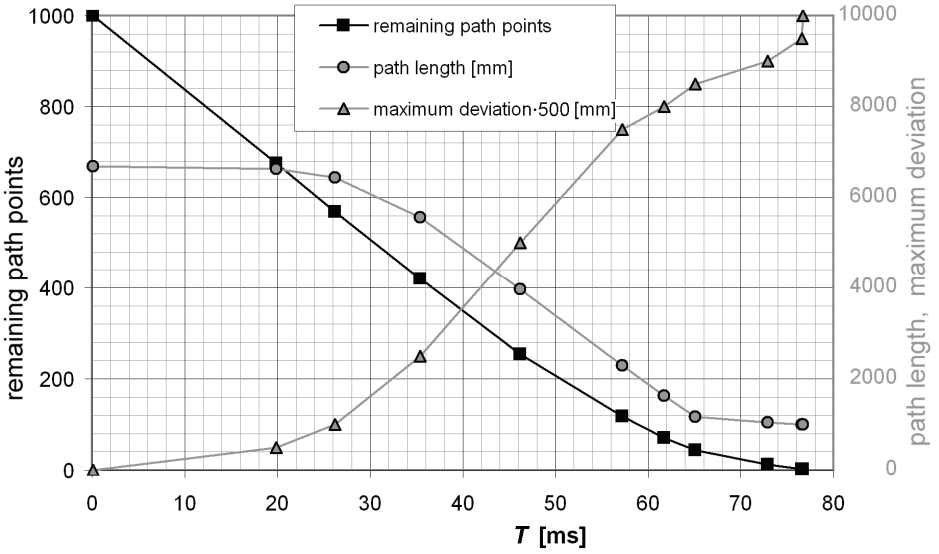

Figure 9. Remaining path points, path length and maximum deviation against computation time T [ms] for the perturbed linear path segment of Figure 8

When milling cavities in workpieces, problems with overly fragmented and angulated milling paths arise, cf. the RONAF project (Robot-based Navigation for Milling at the Lateral Skull Base (Federspil, P. A.; Geisthoff, U. W.; Henrich, D. & Plinkert, P. K., 2003)) (Figure 10). The goal of the RONAF project is the development and examination of a system for navigation on the lateral skull base with the purpose of an interactive supervision of a surgical robot during interventions. Modular navigation and control procedures are being used. The operation is planned on a preoperatively acquired 3D dataset, e.g. computed tomography (CT) or magnetic resonance tomography (MRT). The robot and its attached tool are moved relative to this dataset.

Milling in the skull bone demands high precision (with tolerances below one millimeter) in spite of the high force required to remove larger quantities of bone, a combination that is very straining for a surgeon and poses little problem to a robot. Therefore an important increase of processing quality is expected.

In the RONAF system, three-dimensional path planning (Waringo, M.; Henrich D., 2004) is used in order to mill a given implant volume with a robot-controlled miller. The paths planned in a voxel space are angled and are often represented by an excessive number of path points. The robot follows the path points by interpolating linearly between two successive path points. By reducing the number of path points, we can significantly reduce the milling duration.

Path smoothing was applied to milling paths planned in a voxel space for milling a hearing aid implant volume (Figure 11). The milling time for the non-smoothed path is unsatisfactory long. The traversal speed is strongly reduced in regions where the path points are close to each other or where the directional change between two consecutive path segments is high. This drawback is due to robot dynamics restrictions such as the maximum

acceleration in the robot joints and restrictions involving computing time (i.e. the maximum number of path segments that can be processed per second).

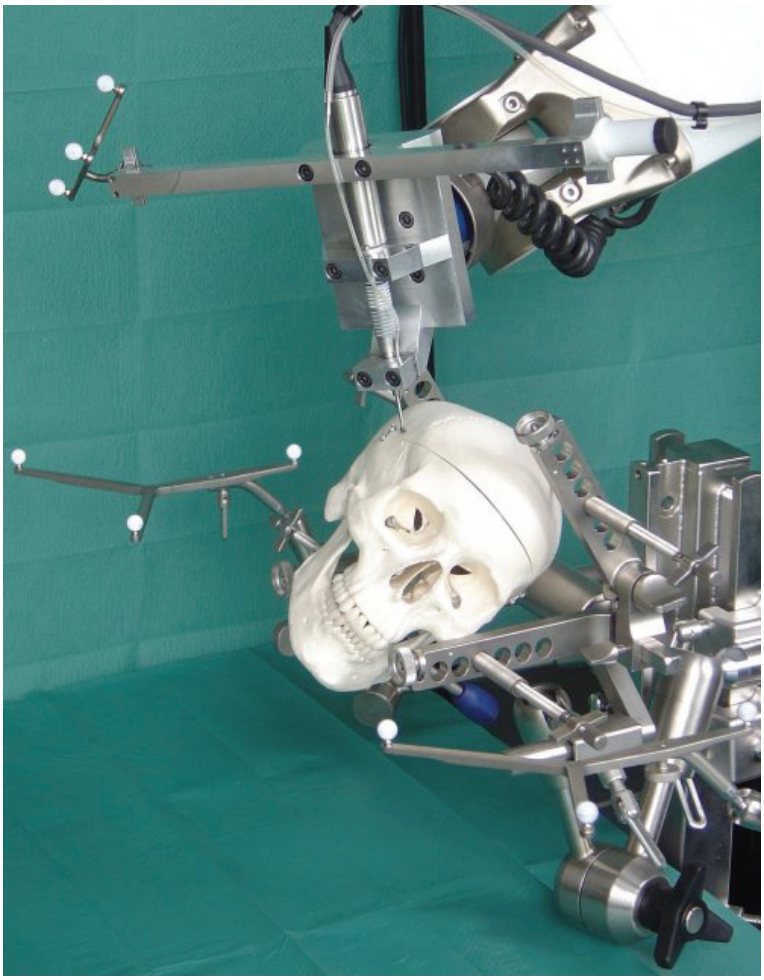

Figure 10. Experimental setup from the RONAF project

Part of the robot motions occurs above the workpiece (the long straight segments in Figure 11), where the tool moves above the bone without touching it. These segments serve to change the currently processed region. They segments can not be removed, since otherwise the miller would mill bone at forbidden locations. Therefore, even with a maximum allowed deviation of infinite in the path smoothing process, such a milling path can not be reduced to its start and end point. The path smoothing only applies to the horizontal path segments located in the bone. The smoothing algorithm was applied to the entire path, with all vertical and horizontal path segments needed for changing a region marked as non-removable.

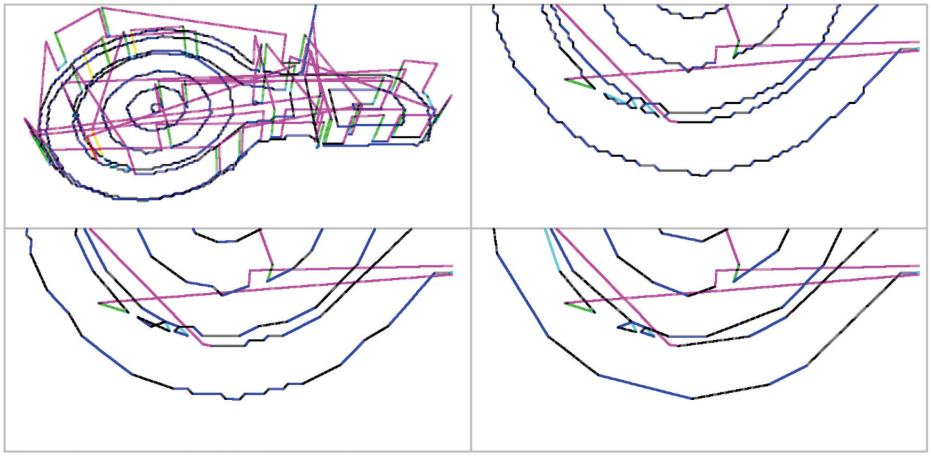

Figure 11. Milling paths for the hearing aid implant Vibrant Soundbridge (Siemens/Symphonix). The circular paths lie in the horizontal plane, and path segments perpendicular to that plane are vertical segments which connect these circular paths. Upper left: original path planned in a voxel space; Upper right: close-up of the original path (4403 path points); Bottom left: path with maximum deviation of 0.18 mm (2788 path points); Lower right: path with maximum deviation of 0.35 mm (1405 path points)

While keeping the modification to the path at a non-critical level so that no noticeable changes occur in the milled geometry [2], the number of path points can be reduced by more than 50%, as shown in Table 3. With an acceptable tolerance of 0.35 mm, it is possible to eliminate about 46% of the milling duration, 69% of the path points and 65% of all changes normally arising in the non-smoothed path. The computation time for path smoothing was measured on an AMD Athlon XP 2600+ PC with 512 MB of RAM. The computation time is nearly exactly linear with the number of points removed, in this example about 0.6 ms per point.

Table 4 shows a comparison of paths smoothed using the three error functions and evaluated according to the three error functions. As expected, path planned with error function K_i, $i \in \{1,2,3\}$ ranked best when the evaluation was performed according to the same error function K_i. No clear advantage can be determined and no error function is made redundant by the other two.

[2] As the miller's diameter is 4.5 mm and the robot's repeatability accuracy is 0.35 mm, a maximum deviation in the path of 0.35 mm is acceptable.

maximum deviation d_{lim} [mm]	# path points	time required for path traversal [min:sec]	computation time [sec]	path length [mm]	angular integral [°]
0	4403	12:35	0.00	5545	239,417
0.10	4355	12:24	0.05	5527	234,048
0.13	2788	09:22	0.94	5460	150,501
0.18	2107	08:06	1.32	5427	118,268
0.25	1629	07:12	1.58	5403	96,645
0.35	1405	06:44	1.77	5367	82,650
0.60	1207	06:19	1.80	5334	74,520
1.00	1098	06:05	1.88	5306	72,273
2.00	997	05:49	1.90	5232	69,016
Infinite	832	04:21	2.04	4171	58,320

Table 3. Number of path points remaining, necessary time requirement for traversal and for path smoothing, path length and angular integral in the function of the maximum allowed deviation. In order to avoid damage to the patient, areas of the path are not allowed to be modified, as described previously. Therefore, the path can not be reduced to its start and end point with an infinite maximum deviation. Without this restriction, the effects of the path shortening would be even more noticeable

		Error function for path evaluation					
		K_1 [mm]		K_2 [mm²]		K_3 [mm²]	
		max	avg.	max.	avg.	max.	avg.
Error function for path computation	K_1	0.281	0.171	0.078	0.015	2.125	0.257
	K_2	0.478	0.216	0.046	0.019	1.531	0.358
	K_3	0.884	0.197	0.297	0.020	0.393	0.213

Table 4. Comparison of the three error functions K_1, K_2 and K_3 when reducing the milling path of the implant Vibrant Soundbridge from 4403 to 1500 path points. K_1: maximum deviation, K_2: root-mean-square deviation, K_3: spanned area. The error function used for path planning is noted in the rows and the error function used for evaluation is noted in the columns. In the cells, the deviations are noted, with both the maximum and the average value per path segment

Another example of path smoothing in the RONAF project is shown in Figure 12. In order to record an ultrasound image of the patient's skull, the skull is sampled manually with the tip of an infrared marker whose spatial position is sampled at equidistant times. This path is then traversed by the robot, an ultrasound head being mounted on the effector. Between recording and traversing of the path, smoothing is performed. This way, in addition to rendering the path less jerky, there are also agglomerations of path points being removed which appear when the surgeon interrupts the movement of the marker.

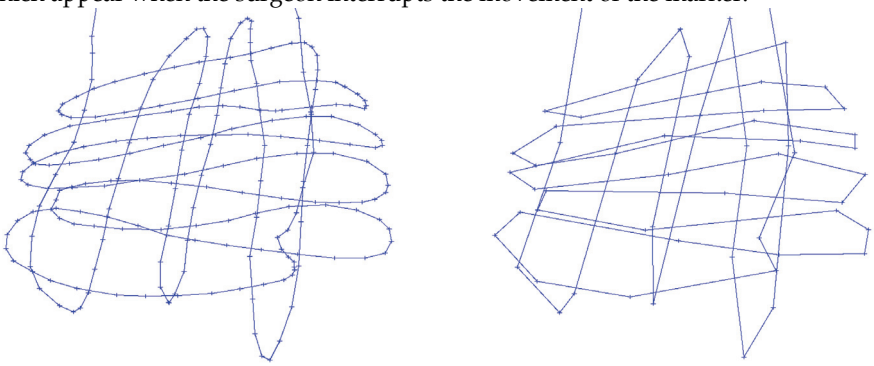

Figure 12. Scanline path for ultrasound recording of the human skull. Left: original path (307 path points), right: smoothed path using K_1 and d_{lim} = 1 mm (90 path points)

8. Conclusions

We have presented a method that smooths paths of any dimensionality consisting of linear segments until the deviation between the smoothed path and the original path locally exceeds a given threshold. The error function for deviation determination can easily be exchanged and adapted for diverse applications. The computational requirement has been reduced from quadratic to linear in the number of path points used. Our method is anytime-capable, i.e. it can be aborted at any time and returns a valid path for which the maximum deviation increases monotonically and the number of path points decreases monotonically in the allowed computation time.

Possible extensions of the algorithm include the consideration of forbidden regions that may not be crossed by the path and a distance computation that varies depending upon the position on the path. Additionally, if applied to motion planning in a cluttered environment, the algorithm does not handle collisions with obstacles close to the unsmoothed path. In this case, further conditions are required which are evaluated in addition to the error functions and which avoid the smoothed path getting too close to the obstacles. In this scenario, the geometry of the actuator must be considered too.

For a path smoothing with using both positions and orientations as optimization criterion (Case (e) in Section 7), one could in addition use one of them as constraint, giving even more control over the maximum allowed deviation.

If a globally optimal path has to be found, the presented method is not suitable, as it is a local method and can get stuck in local optima, as shown in the first experiment. In order to overcome this disadvantage, a global method has to be used.

9. Acknowledgments

This work has been supported by the German Research Foundation (DFG) under the project name "Robotergestützte Navigation zum Fräsen an der lateralen Schädelbasis" (RONAF) with the identifier 227100. Further information can be found at *http://ai3.inf.uni-bayreuth.de/projects/ronaf*.

10. References

Aleotti, J.; Caselli, S. (2005). Trajectory clustering and stochastic approximation for robot programming by demonstration, *IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, pp. 2581-2586

Amato, N.M. & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning, *Proceedings of the IEEE Int. Conf. Robot. & Autom.*, pp. 113-120.

Baginski, B. (1998). Motion planning for Manipulators with Many Degrees of Freedom – The BB Method, *Doktorarbeit*, TU München

Barraquand, J. & Latombe, J.-C. (1991). Robot motion planning: a distributed representation approach, *Int. Journal of Robotics Research*, 10 (6), pp. 628-649, 1991

Berchtold, S. & Glavina, B. (1994). Kosten-Nutzen-optimale Verbesserung kollisionsfreier Roboterbewegungen mittels Polygon-Manipulation, in: *10. Fachgespräch Autonome mobile Systeme*

Bronstein, I. N.; Semendjajew, K. A; Musiol, G.; Mühlig, H. (2001). *Taschenbuch der Mathematik*, Verlag Harri Deutsch, ISBN 3-8171-2005-2

Carpin, S.; Pillonetto G. (2006). Motion planning using adaptive random walks, *IEEE Transactions on Robotics and Automation*, 21 (1)

Engel, D. (2003). Sensorgestützte Robotersteuerung für den Einsatz in der Chirurgie, Dissertation, *Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (TH)*

Federspil, P. A.; Geisthoff, U. W.; Henrich, D. & Plinkert, P. K. (2003). Development of the First Force-Controlled Robot for Otoneurosurgery, *Laryngoscope* 113

Geraerts, R. & Overmars, M. H. (2002). A comparative study of probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*

Hein, B. (2003). Automatische offline Programmierung von Industrierobotern in der virtuellen Welt, Dissertation, *Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (TH)*, 2003.

Kuffner, J. J.; LaValle S. M. (2001). RRT-Connect: An efficient approach to single-query path planning, in *Proceedings of the IEEE Conference on Robotics and Automation*, San Francisco, pp. 995–1001.

Maillot, P. (1990). Using quaternions for coding 3d transformations, in *Graphics Gems I*, Academic Press Inc., Boston pp. 498-515

Quinlan, S.; Khatib, Q. (1993). Elastic bands: Connecting Path Planning and Control, in *Proceedings of the IEEE Conference on Robotics and Automation*, Atlanta, pp. 802-807.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning, TR 98-11, *Computer Science Dept., Iowa State University*

Urbanczik, C. (2003). SIMERO: Bildbasierte Kollisionserkennung und Bahnglättung im Konfigurationsraum, Diploma Thesis, *Fakultät für Informatik, Universität Kaiserslautern*, D-67653 Kaiserslautern

Waringo, M.; Henrich D. (2004). 3-Dimensionale schichtweise Bahnplanung für Any-Time-Fräsanwendungen, *Robotik*, München/Germany

Waringo, M.; Henrich D. (2006). Efficient smoothing of piecewise linear paths with minimal deviation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing/China, pp. 3867-3872