IROS 2006, IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing/China, 9.-15. 10. 2006, pp. 3867-3872

Efficient Smoothing of Piecewise Linear Paths with Minimal Deviation

Michel WARINGO and Dominik HENRICH

Lehrstuhl für Angewandte Informatik III (Robotik und Eingebettete Systeme)
Universität Bayreuth, D-95445 Bayreuth, Germany

{michel.waringo, dominik.henrich}@uni-bayreuth.de

Abstract - **We present an anytime fast deterministic greedy method for smoothing piecewise linear paths consisting of connected linear segments by incrementally removing path points. First, points with only a small influence on path geometry are removed, i.e. aligned or nearly aligned points. Due to the removal of less important path points, the high computational and storage space requirements of the paths are reduced and traversal of the path is accelerated. Our algorithm can be used in the most diverse applications, i.e. sweeping, path finding or programming-by-demonstration in a virtual environment.**

Index Terms – linear path, smoothing, shortening, merging, thinning, anytime, deviation

I. INTRODUCTION

In the area of robotics, the pointwise traversal of a given path is a popular task, e.g. in mobile, industrial or surgical robotics. Describing paths by a sequence of linear segments is the easiest method, and for many tasks the precision of a path approximated by linear segments is sufficient. The movements to be accomplished with a mobile robot or the robot's endeffector, e.g. a gripper mounted on the last link, are described by a sequence of points in space that are to be traversed by the robot. 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 desired path, leading to unnecessary mechanical stress of both robot and tool. Furthermore, 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 often consists of many path points. In this method, 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 is subject to 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 forms approximation of diagonal lines. 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 II), after which the state of the art is presented (Section III). Then, the proposed procedure is described in detail (Section IV), and different specializations of the proposed method are pointed out (Section V). Finally, experiments are described (Section VI), and open issues and further enhancements are discussed (Section VII).

II. PROBLEM DESCRIPTION

The problem of path smoothing can be described as follows. A *path* $P = \langle p_1, p_2, \ldots , p_n \rangle$ is given, represented by an ordered list of m-dimensional Cartesian *path points* $p_i = (p_{i1}\ p_{i2} \ldots p_{im})^T$. All path points p_i have the same dimensionality m, which can vary depending on the degrees of freedom of the robot or the requirements of the task to be performed.

The *neighbourhood* of a path point p_i is defined as the sequence of points in P between and including the two nearest neighbours of that point in the path P to the left and and right of p_i. The *deviation* p_i.d [1] between the smoothed path P' and the original path P in the neighbourhood of the path point p_i can be computed according to various error functions, such as the standard deviation or the area spanning both paths.

The *error function K* represents the criterion used to compute p_i.d. Its input are the two paths P and P' as well as the index i, and its output is the deviation p_i.d between them. Finally, we need a threshold value d_s indicating the maximum allowed p_i.d. The entire smoothing process uses the same K.

Thus, we search for a path P' whose deviation from P does not exceed d_s at each individual path point according to K. The number of path points of the path P' is minimized under the given conditions (Fig. 1).

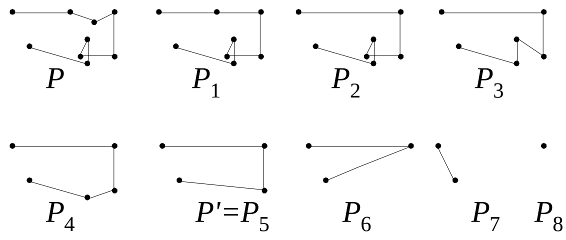

Fig. 1: Example of complete smoothing of a two-dimensional path P with nine path points. In each step i, the path point whose removal leads to the smallest possible deviation between P_i and the original path P is removed. A reasonable smoothed path could be e.g. P_5.

[1] The variable d can be conceived as being part of the data structure representing the point p_i

III. State of the Art

We can distinguish two main categories of problems where a reduced number of path is sought: path smoothing and path shortening.

In collision-free motion planning, e.g. for mobile robots, the start and the end point are given, and a path connecting them is sought. There may be obstacles and constrictions like doors or corridors. A good path is short and avoids all obstacles. In a first step, e.g. a randomized approach, a path of poor quality is created, containing many useless segments and being much longer than necessary. It is improved in a second step by a *path shortening* process (Subsection A). There is no need for any similarity between the unshortened and the collision-free shortened path apart from 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 provided. However, the quality of this path may be unsatisfying, being jerky or consisting of too many segments. In this *path smoothing* process (Subsection B), small deviations from the original path are allowed as long as the number of path segments can be reduced notably and the similarity to the original path stays high enough.

A. Path shortening processes

In collision-free motion planning, planning is performed usually in the configuration space. The problem of finding a path between a start and an end point is PSPACE-hard in the degree 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 [10, 2] proceed in two steps. First, a collision-free path is constructed as a graph in configuration space. In a second step, pairs of promising vertices are chosen and a simple local planner is used to find a feasible connection between them. Approaches based on Rapidly-exploring Random Trees (RRTs) [14, 12] 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.

There exist other path planning algorithms 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 not too many obstacles in the configuration space. One example is the Randomized Path Planner (RPP) [4] 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 are based on elastic-band method [13], where contractive and repulsive forces emanating from obstacles determine the deformation of an original path.

When the curve 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 [3]. A similar procedure is used in [5]. Based on a heuristic, path points are removed to locally reduce the path length. Another example of collision-free paths for robots can be found in [15]. 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 smoothing 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 [7]. However, these approaches are not valid for the application we envisage because a similarity between the original and the smoothed path can not be guaranteed.

B. Path smoothing processes

In approaches 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 reached by approximating the path by curves of a higher degree consisting of nonlinear path segments (e.g., defined by quadratic or cubic functions) [11] or non-uniform rational B-Splines (NURBS) [1].

In [8] a smoothing procedure for piecewise linear paths is described that removes path points p_j not exceeding a given deviation from a path segment $<p_i, p_k>$ 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 smoothing a path already smoothed in a previous step.

A well-known point reduction method is linear regression [6], 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.

C. Conclusion

One can conclude that there is a need for smoothing paths in diverse robotic applications. 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.

The path smoothing method we propose offers some advantages which make it particularly suited for time-critical applications working either in configuration space or task space. It 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 exchanged easily 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.

IV. Path smoothing procedure

The path points p_i are stored in a list ordered by index. Their description (i.e., the Cartesian coordinates) is extended by two components:

- The flag $p_i.\mathrm{r} \in \{true,\ false\}$ indicates whether the path point has been removed while smoothing.
- The variable $p_i.\mathrm{d} \in \mathrm{R}$ indicates the deviation of the path in the neighbourhood of the path point, which will be explained in detail in Section V.

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 second 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 p_k not yet removed ($p_k.\mathrm{r} = false$), compute the arising deviation $p_k.\mathrm{d}$ between the smoothed path P' and the original path P, assuming that p_k is removed from the path (in addition to the path points already removed so far).

(2) Select the path point p_k with the smallest $p_k.\mathrm{d}$.

(3) If the deviation $p_\mathrm{k}.\mathrm{d}$ is smaller than the specified value d_s, then mark p_k as removed ($p_\mathrm{k}.\mathrm{r} = true$) and go to (1).

(4) Otherwise, no further path points can be removed from the path, and the path $P' = < p_i \mid p_i.\mathrm{r} = false >,\ i = 1, \ldots, n$ is returned.

The path point whose elimination leads to the smallest deviation from the original path is removed during each iteration. In this way, it is ensured that a (local) maximum number of path points can be removed before the deviation locally exceeds the threshold d_s and the algorithm terminates.

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 optimization can be obtained by using a simple trick. Given a path with n path points, the maximal smoothing of the path would require n–2 iterations, as the first and last path point are not removed. For a given iteration step i, the number of local deviation computations is n–2–i. This belongs to the complexity class O(n^2), since the total number of computation steps is

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

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

The procedure can be accelerated by considering that the path deviation only changes in the neighbourhood of a path point that is removed. For this purpose, the component $p_i.\mathrm{d}$ of all path points p_i is required to buffer the corresponding deviation.

After a path point has been removed, the deviation can change only for the two neighbouring path points. Therefore, only two instead of n–2–i deviations need to be determined per iteration step. This results in a complexity of

$$n + \sum_{i=2}^{n-2} 2 \;=\; n + 2(n-3) = 3n - 6$$

computation steps, complexity dropping from O (n^2) to O (n).

Prior to each iteration step, the deviation $p_i.\mathrm{d}$ is known for all remaining path points p_i not yet removed (thus all path points with $p_i.\mathrm{r} = false$). Based on this information, the path point whose removal leads to the smallest deviation from the original path can reliably be removed.

In the following we describe how the interval of path points is determined that is needed for the computation of the deviation $p_i.\mathrm{d}$. We are looking for two path points p_min and p_max that border on the interval in question: $I = [p_\mathrm{min}\,;\ p_\mathrm{max}] = < p_\mathrm{min}, \ldots, p_i, \ldots, p_\mathrm{max} >$.

In the first iteration no path points have been removed yet. Trivially, only three path points need to be regarded: the path point p_i as well as its neighbours $p_\mathrm{i-1}$ and $p_\mathrm{i+1}$, and the path segment $< p_\mathrm{i-1},\ p_\mathrm{i},\ p_\mathrm{i+1} >$ must be compared with $< p_\mathrm{i-1}, p_\mathrm{i+1} >$[2]. The manner in which this comparison is performed depends on the error function used and is described in Section V.

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 p_min and p_max. Thus p_min und p_max are the direct neighbours of p_i that have not yet been removed. The deviation $p_i.\mathrm{d}$ is computed by comparing a path segment of the original path $P_{i.\mathrm{o}} = [p_\mathrm{min}, p_\mathrm{max}] = < p_\mathrm{min}, \ldots, p_i, \ldots, p_\mathrm{max} >$ and the corresponding path segment on the smoothed path $P_{i.\mathrm{t}} = < p_\mathrm{min}, p_\mathrm{max} >$.

V. Path deviation functions

Another important component of the presented algorithm is the computation of the deviations $p_i.\mathrm{d}$. Depending on the application, different error functions K can be used. We investigated three error functions:

- K_1: $p_i.\mathrm{d}$ is the maximum Euclidean deviation of the smoothed path from the original path in the interval $P_{i.\mathrm{o}}$. 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 endoscopic holding systems.
- K_2: $p_i.\mathrm{d}$ is the root-mean-square deviation of the path points in the original path $P_\mathrm{i.o}$ from the shortened path $P_{i.\mathrm{t}}$ in the interval $P_{i.\mathrm{o}}$. This is the square root of the mean of the squares of the shortest distances between each of the path points $p_i \in P_{i.\mathrm{o}}$ and the segment $P_{i.\mathrm{t}}$. This criterion is useful mainly for theoretical analysis.
- K_3: $p_i.\mathrm{d}$ is the area between the smoothed paths segment $P_{i.\mathrm{t}}$ and the corresponding segment $P_{i.\mathrm{o}}$ on the original path. K_3 is the best choice for sweeping applications, e.g. bones milling or cleaning robots.

[2] This computation optimization is valid because the path segments are linear.

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_s from the original path. One disadvantage involves the computation of each deviation p_i.d: Only one path point $p_i \in [p_{\min} ; p_{\max}]$ is used and the distance from the other path points in that interval is neglected. Path points far away from $P_{i.t}$ 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 $p_i \in [p_{\min} ; p_{\max}]$ for the computation. Nevertheless, path points far away from $P_{i.t}$ are recognized because the distance to the smoothed path $P_{i.t}$ 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 the individual path points $p_i \in [p_{\min} ; p_{\max}]$, not just in relation to $P_{i.t}$.

Fig. 2 illustrates the three error functions.

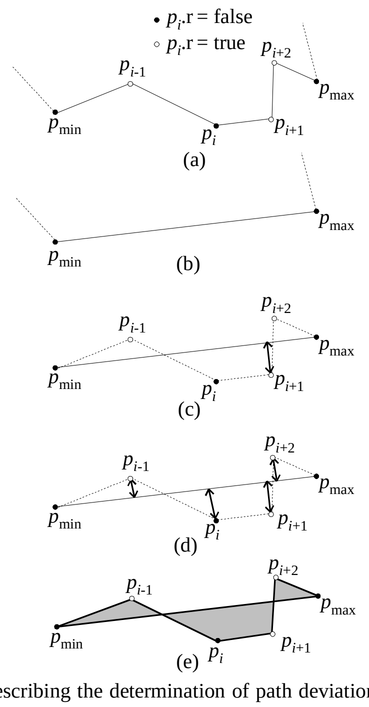

Fig. 2: Sketch describing the determination of path deviation p_i.d. The linear path segments to be compared are the original path (a) and the smoothed path (b). The error functions K_1, K_2, and K_3 are illustrated in (c), (d), and (e)

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 in any iteration step, opportunities to shorten the path can be missed. Consider for example Fig. 3 (a). When using criterion K_1 and a maximum allowed deviation $d_s = 0.5 \cdot \| p_2\, p_3 \|$ (half the distance between p_2 and p_3), the optimal path (b) cannot be obtained. The removal of either p_2 or p_3 leads to a deviation that is too high. The smoothing process aborts without being able to remove p_2 and p_3.

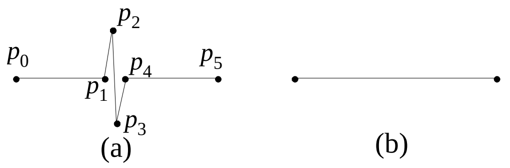

Fig. 3: Example for the non-optimality of the proposed algorithm. (a) A path consisting of five points, (b) the optimal path with a maximum allowed deviation of $0.5 \cdot \| p_2\, p_3 \|$

Nevertheless, the paths created are valid because the deviation does not exceed the maximum allowed one moment. 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.

VI. Experiments and Results

In this section, we present and analyze a practical surgical application of our algorithm. First, we briefly describe the original paths used in the surgical application and their drawbacks. Then we describe the improvement achieved by applying the smoothing algorithm. Finally, we compare the results yielded when applying the three error functions described previously.

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 [9]). 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. CT or MRT). The robot and its attached tool are moved relative to this dataset. Milling in the skull bone demands extreme precision (with tolerances well 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 can be expected.

In the RONAF system, three-dimensional path planning [16] is used in order to mill out 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 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 (Fig. 4). 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).

Part of the robot motions occurs above the workpiece (the long straight segments in Fig. 4), where the tool moves above the bone without touching it. These segments serve to change the currently processed region. These path segments cannot be removed, since otherwise the miller would mill bone at invalid locations. 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.

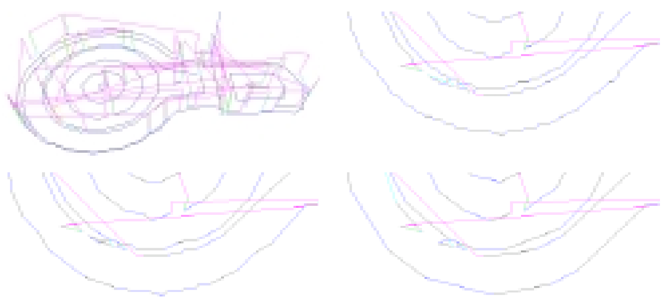

Fig. 4: Milling paths for the hearing aid implant Vibrant Soundbridge (Siemens/Symphonix). The circular paths lie is the horizontal plane, and path segments perpendicular to that plane are vertical segments. Upper left: original path planned in a voxel space, upper right: close-up of the original path (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[3], the number of path points can be reduced by more than 50%, as shown in Table 1. The permissible path deviations are low enough so that no noticeable changes occur in the milled geometry.

Table 1 shows some experimental results. 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 directional 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 2 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.

[3] 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 deemed acceptable.

Table 1: 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.

maximum deviation d_s [mm]	# path points	time req. for path traversal [min:sec]	comp. 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 2: 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: integral of the 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.

		Error function for path evaluation		
		K_1	K_2	K_3
Error function for path com-	K_1	max : 0,281 mm avg: 0,171 mm	max: 0,078 mm^2 avg: 0,015 mm^2	max : 2,125 mm^2 avg: 0,257 mm^2
	K_2	max: 0,478 mm avg: 0,216 mm	max: 0,046 mm^2 avg: 0,019 mm^2	max : 1,531 mm^2 avg: 0,358 mm^2
	K_3	max: 0,884 mm avg: 0,197 mm	max: 0,297 mm^2 avg: 0,020 mm^2	max : 0,393 mm^2 avg: 0,213 mm^2

Another example of path smoothing in the RONAF project is shown in Fig. 5. 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.

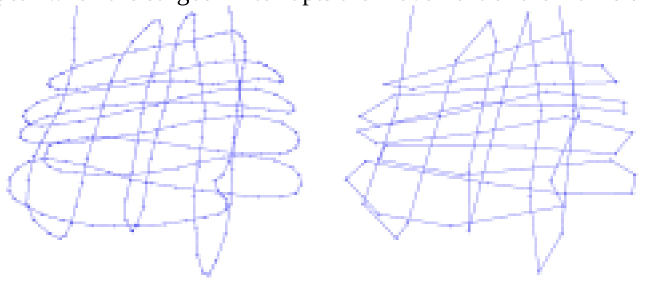

Fig. 5: Scanline path for ultrasound recording of the human skull. Left: original path (307 path points), right: smoothed path using K_1 and d_s= 1 mm (90 path points).

In the next 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. Fig. 6 shows the results of the path smoothing.

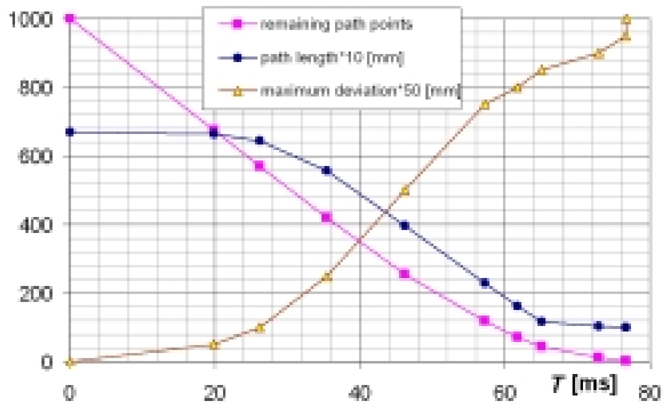

Fig. 6: Remaining path points, path length and maximum deviation against computation time T[ms] for a perturbed linear path segment

This experiment demonstrates the nice anytime property of the algorithm. It can be aborted at any moment. 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 non-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_s=10 mm. Yet, the algorithm finds that solution only at d_s=20 mm in general.

VII. Conclusion and Outlook

We have developed 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. Thanks to the additional variable p_i.d and flag p_i.r used for each path point, the computational requirement is reduced asymptotically. The experiments show that smoothing the path brings about considerable advantages in our application.

Our method is anytime-capable, i.e. it can be aborted at any moment and returns a valid resulting path for which the maximum deviation increases monotonically and the number of path points decreases monotonically in time. There is no need for an additional increase in computation speed and the storage requirement is also not critical, as it is linear with respect to the number of path points.

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.

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 effector must be considered too.

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.*

References

[1] J. Aleotti, S. Caselli: “Trajectory clustering and stochastic approximation for robot programming by demonstration”, 2005 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems, pp. 2581-2586.

[2] N.M. Amato, Y. Wu: “A randomized roadmap method for path and manipulation planning” , Proceedings of the 1996 IEEE Int. Conf. Robot. & Autom., pages 113-120

[3] B. Baginski: „Motion planning for Manipulators with Many Degrees of Freedom – The BB Method“, Doktorarbeit, TU München, 1998

[4] J. Barraquand, J.C. Latombe: “Robot motion planning: a distributed representation approach”, Int. Journal of Robotics Research, 10 (6), pages 628-649, 1991

[5] S. Berchtold, B. Glavina: „Kosten-Nutzen-optimale Verbesserung kollisionsfreier Roboterbewegungen mittels Polygon-Manipulation“, in: 10. Fachgespräch Autonome mobile Systeme, 1994

[6] I. N. Bronstein, K. A. Semendjajew, G. Musiol, H. Mühlig: „Taschenbuch der Mathematik“, Verlag Harri Deutsch, 2001, ISBN 3-8171-2005-2

[7] S. Carpin, G. Pillonetto: “Motion planning using adaptive random walks”, IEEE Transactions on Robotics and Automation, 21 (1), 2006

[8] D. Engel: „Sensorgestützte Robotersteuerung für den Einsatz in der Chirurgie“, Dissertation, Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (TH), 2003

[9] P. A. Federspil , U. W. Geisthoff , D. Henrich , P. K. Plinkert: „Development of the First Force-Controlled Robot for Otoneurosurgery“, Laryngoscope 113: March 2003

[10] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR), December 2002

[11] B. Hein: „Automatische offline Programmierung von Industrierobotern in der virtuellen Welt“, Dissertation, Fakultät für Informatik der Universität Fridericiana zu Karlsruhe (TH), 2003.

[12] J.J. Kufner and S.M. LaValle, “RRT-Connect: An efficient approach to single-query path planning,” in Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, April 2001, pp. 995–1001.

[13] S. Quinlan, Q. Khatib: „Elastic bands: Connecting Path Planning and Control”, in Proceedings of the IEEE Conference on Robotics and Automation, Atlanta, 1993, pp. 802-807.

[14] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[15] C. Urbanczik: „SIMERO: Bildbasierte Kollisionserkennung und Bahnglättung im Konfigurationsraum“, Diplomarbeit, Fakultät für Informatik, Universität Kaiserslautern, D-67653 Kaiserslautern, 2003

[16] M. Waringo: „3-Dimensionale schichtweise Bahnplanung für Any-Time-Fräsanwendungen “, Robotik 2004, München/Germany