Efficient Error-bounded Curvature Optimization for Smooth Machining Paths

Evgenia Selinger* and Lars Linsen*‡

*Department of Computer Science and Electrical Engineering
Jacobs University, Bremen, Germany
‡Department of Mathematics and Computer Science
Westfälische Wilhelms-Universität Münster, Germany
email: linsen@uni-muenster.de

Abstract

Automated machining with 3-axis robots requires the generation of tool paths in form of positions of the tool tip. For 5-axis robots, the orientations of the tool at each position needs to be provided, as well. Such a tool path can be described in form of two curves, one for the positional information (as for 3-axis machining) and one for the orientational information, where the orientation is given by the vector that points from a point on the orientation curve to the respective point on the position curve. As the robots need to slow down for sharp turns, i.e., high curvatures in the tool path lead to slow processing, our goal is to generate tool paths with minimized curvatures and a guaranteed error bound. Starting from an initial tool path, which is given in the form of polygonal representations of the position and orientation curves, we generate optimized versions of the curves in the form of B-spline curves that lie within some error bounds of the input path. Our approach first computes an optimized version of the position curve within a tolerance band of the input curve. The outcome of this first step can directly be applied to 3-axis machining. Based on this first step, for 5-axis machining the orientation curve needs to be updated to again fit the position curve. Then, the orientation curve is optimized using a similar approach as for the position curve, but the error bounds are given in the form of tolerance frustums that define the tolerance in lead and tilt. For an efficient optimization procedure, our approach analyzes the input path and splits it into small (partially overlapping) groups before optimizing the position curve. The groups are categorized according to their geometric complexity and handled accordingly using two different optimization procedures. The simpler, but faster algorithm uses a local spline approximation, while the slower, but better algorithm uses a local sleeve approach. These algorithms are adapted to both the position and orientation curve optimization. Subsequently, the groups are combined into a complete tool path in the form of G^2-continuous B-spline curves, where we have one such curve for 3-axis machining and two such curves defined over the same knot vector for 5-axis machining.

First presented at the International Conference on Computer Graphics Theory and Applications 2018, extended and revised for JVRB

Keywords: NURBS curves, 3-axis and 5-axis Machining, G^2- continuity.

1 Introduction

The sculptured surface machining technology has become a widely used technology in manufacturing engineering, e.g., for shaping metal and other solid materials. A necessary integral part is the generation of a milling path. The milling machine cuts along a given path to obtain the required shape of the workpiece.

DOI 10.20385/1860-2037/15.2018.2, ISSN 1860-2037

For 5-axis milling, the cutter rotates around the spindle axis, the spindle can move in all three spatial directions, and the workpiece is fixed to a table, which can also move in two angular directions to allow the milling tool to have access to the workpiece from different orientations. The flexibility of 5-axis machining allows for manufacturing of more complicated workpieces and for higher quality. However, it requires the control of the five axes. The first three axes represent the position of the milling tool in form of *xyz*-coordinates, the other two are necessary to represent the orientation of the table to which the workpiece is attached, i.e., the angle between the milling tool and the table (see Figure 1a).

The tool path for 5-axis machining is given in form of a position curve of the tool tip and a respective orientation of the tool for each point on the position curve. We assume that the orientation is given in form of an orientation curve, where the implied orientation is given by the vector that points from a point on the orientation curve to the respective point on the position curve. As sudden changes in the tool path requires the robot to slow down to perform the respective movements, it is desirable to have tool paths with low curvature on both position and orientation curve. Our goal is to minimize the overall manufacturing time, i.e., to generate a curvature-optimized tool path for 5-axis machining. Our algorithms take a given tool path and performs an optimization of position and orientation within given error bounds. As this optimization is embedded into the manufacturing workflow, it needs to be efficient, as well.

The input to our algorithm is a sequence of pairs (p,q) of points. The points p are the positions of the milling tool on the workpiece, the points q are the orientation points of the milling tool at that time. When connecting the sequences of points p and q, respectively, they describe the position and orientation curves in polygonal representation. The orientation vector $q-p$ given as the difference of the position and the orientation point can be directed at various angles to the workpiece depending on the desired manufacturing process. The angle is defined by values of lead and tilt. Lead is the angle between the normal vector N of the workpiece's surface and the orientation vector $q-p$ of the milling tool in the direction of movement F. Tilt is the angle between the normal vector N and the orientation vector $q-p$ of the milling tool perpendicular to the direction of movement F (see Figure 1b).

Our optimization procedure starts with the optimization of the position curve. The curvature of the curve shall be minimized within a given tolerance band. This part of our approach is equivalent to the optimization for 3-axis machining [SL11]. Then, we need to synchronize the representation of the orientation curve with the updated (i.e., optimized) representation of the position curve. Subsequently, we perform the optimization of the orientation curve. To get a desired result we use the restriction that corresponding position and orientation control points have to form an orientation vector in the same direction and length as neighboring initial vectors with some small given allowed deviation of lead and tilt (see Figure 2). For the synchronization of the orientation curve with the optimized position curve, we need to build new orientation vectors starting at the control points of the optimized position curve with the direction depending on the direction of the given neighboring orientation vectors with weights proportional to the distance to them. Based on the given tolerances of lead and tilt, we built frustums around generated orientation points and restrict them to lie within these frustums.

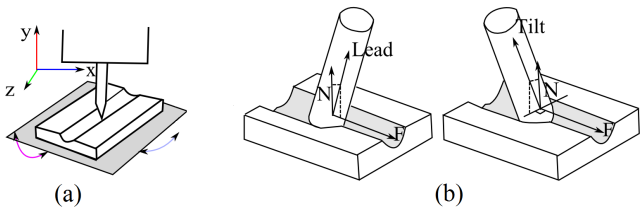

Figure 1: (a) 5-axis machining. (b) Orientational information of the tool in form of lead and tilt

As the optimization of the tool path is embedded in the manufacturing process, it shall be efficient. We propose a method that is based on local optimizations, which are faster to compute than global computations. Therefore, we first split the toolpath into groups, handle groups individually, and combine the results. The handling of the individual groups is based on how complicated their geometry is. Simple groups can be handled with a simple spline approximation with sufficient quality. Complicated groups are handled using a more complex sleeve algorithm. Section 3 describes the overall processing pipeline, which outlines the remainder of the paper.

The resulting position curve is a B-spline curve

$$\mathbf{C}(u) := \sum_{j=0}^{n} \mathbf{P}_j N_j^d(u), a \le u \le b \qquad (1)$$

where the interval $[a,b]$ can be any, $\mathbf{P}_j$ denotes the j-th

DOI 10.20385/1860-2037/15.2018.2, ISSN 1860-2037

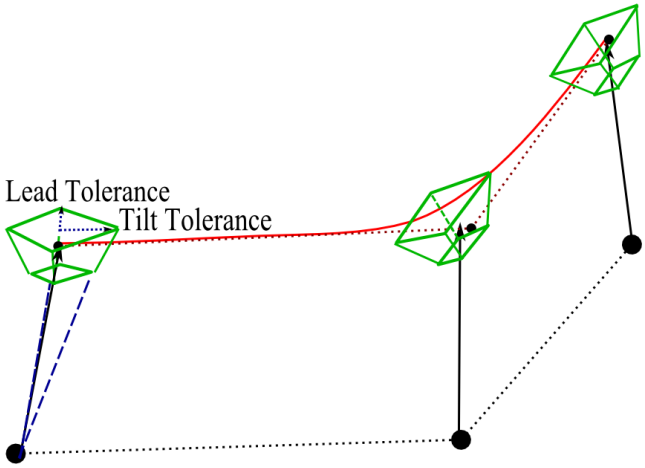

Figure 2: Black vectors are newly generated orientation vectors starting from the optimized position control points. Frustums (green) are built with respect to given tolerances for lead and tilt (blue). Red curve is the resulting B-spline orientation curve with the control polygon shown as dotted dark-red lines.

position control points, and N_j^d denotes the j-th normalized B-spline function of degree d. The B-spline basis functions are defined over a nonperiodic knot vector

$$\mathbf{U} = (\underbrace{a,\cdots,a}_{d+1}, u_{d+1},\cdots,u_{m-d-1}, \underbrace{b,\cdots,b}_{d+1}), \quad (2)$$

where $m = n+d+1$.

The resulting orientation curve shall also be represented as a B-spline curve

$$\mathbf{B}(u) := \sum_{j=0}^{n} \mathbf{Q}_j N_j^d(u), a \le u \le b$$

where $\mathbf{Q}_j$ denotes the j-th orientation control point. The B-spline basis functions N_j^d and knot vector $\mathbf{U}$ are identical to the position basis functions and knot vector.

2 Related Work

A lot of work has been done in 5-axis milling and one of the main topics in this context is to find a proper position and orientation of the milling tool [JCL03]. To make the best use of 5-axis machines they have to solve complicated interference problems and to determine the optimal tool orientations for complex data machining. Gouging and tool collision (see Figure 3) are the main problems in 5-axis machining of sculptured surfaces. Gouging denotes the removal of the excess material in the area of the cutter contact point due to the mismatch in curvatures between the tool tip and part of the milling path. Collision is the interference of the cylindrical part of the tool or tool holder and the workpiece part. The goal is to calculate an optimal slope of the milling tool based on a given workpiece trajectory. We, on the other hand, assume that a tool path is already given. Our goal is to optimize the tool path respect to curvature within an error bound of the given tool path. Some of existing papers [KE02, LLL08] are using kinematic constraints to calculate tool path such as maximum feedrate and acceleration, and also consider the shape of the tool tip, which are not available for us on this step.

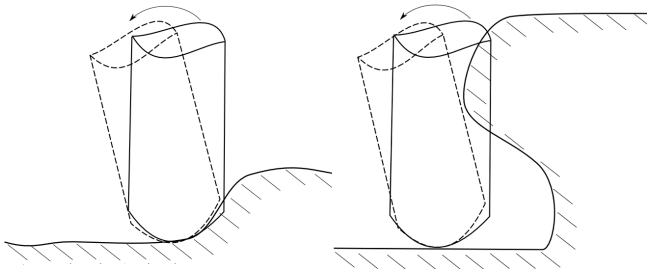

Figure 3: Gouging (left) and collision (right).

In some papers [FS01, LDLB04] it was suggested to find position $P(u)$ and orientation $Q(u)$ B-spline curves by interpolating of position points P_i and orientation points A_i with the restriction that the two B-spline curves satisfy the criterion

$$\exists \mathbf{H} \in R \text{ so that } \forall u, \|Q(u) - P(u)\| = \mathbf{H},$$

where $\mathbf{H}$ is the distance between position and orientation B-spline curves. In our case, we want to get B-spline curves which are generated using approximation technique and we are allowed to have a small given tolerance for $\mathbf{H}$.

The term of double NURBS curve was introduced to define the desired result in industry. It indicates that the output should be presented as two NURBS curves with common knot vector (see [LLYM08, WMCH07]). When such a double NURBS is generated it is decomposed into simple parts understandable by a CNC machine. In our previous work, we [SL11] described a method to obtain curvature-optimized B-spline curves for 3-axis machining (see Section 5 for details). For the 5-axes problem we introduced an approach that is based on the same ideas of local

sleeve approach and local G^2- continuous spline approximation, but extends them to handle 5-axis tool paths [SL18]. In this extended paper of our recent work [SL18], we present a general approach for handling 3- and 5-axis machining toolpath optimizations.

In literature and even textbooks, there exist many approaches for smoothing curves in a general setting, i.e., curves that are not necessarily machining toolpaths. They are based on local smoothing operators or on fitting a smooth curve with some parameters to some noisy data by using some optimization procedure. Such general approaches do typically not consider side constraints such as staying within a given error tolerance that control the deviation from the given curve. Moreover, to our knowledge, there is no approach that handles double NURBS as input. Thus, these methods are not applicable to our problem at hand.

3 Overview

As a part of our approach, we build upon two local approximation algorithms which we use depending on the arrangement of the initial points. The polygonal input curves are split into small groups of point sequences, to which the local algorithms are applied. If one of the angles that are formed by three consecutive points of a group of points are smaller than some threshold α_{sharp}, we consider the group as being complicated and apply the idea of threading splines through 3D channels (see Section 5.1). This algorithm provides us with a local solution with nearly optimal curvature values. However, the optimization involves linear programming methods, which are computationally intense. For the non-complicated (or simple) groups we use the idea of local non rational cubic approximation (see Section 5.2). This algorithm provides acceptable approximation results only for such simple groups, but it is substantially faster than the first algorithm. Afterwards, we combine the locally optimized groups to one global curve. Since an average workpiece consists mostly of simple parts, the computation times of our combined algorithm are significantly lower.

Based on these two local optimization algorithms, we handle the 5-axis tool path optimization by first processing the position curve, then adjusting the orientation curve representation, and finally processing the orientation curve. The overall work flow of our approach is depicted in Figure 4. The individual steps

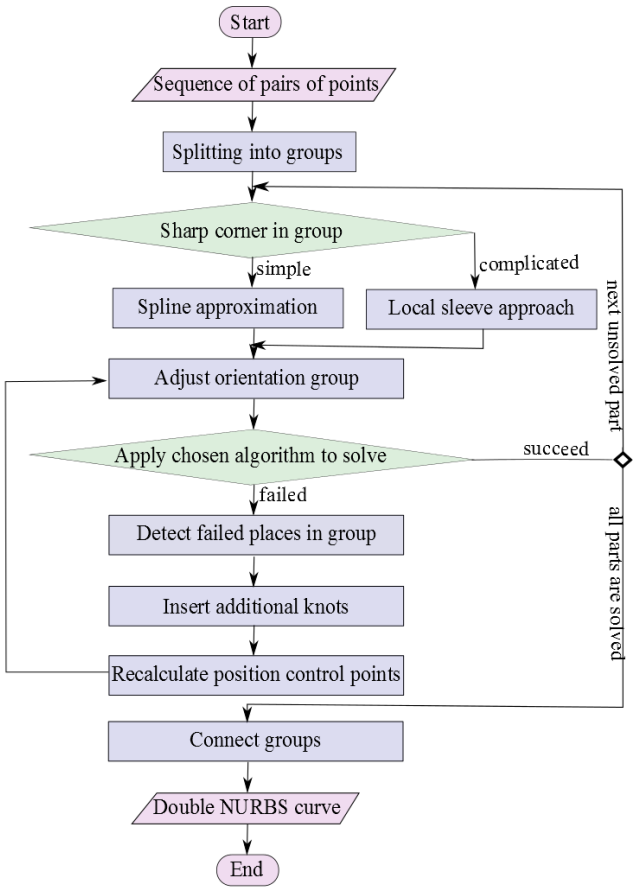

Figure 4: Outline of the process

are described as:

1. Split the sequence of initial position points into small position groups, estimate the groups' properties, and decide which algorithm to apply (see Section 5).

2. For each position group, generate the local solution using the appropriate algorithm, i.e., local sleeve approach for complicated groups or spline approximation for simple groups (see Section 5).

3. For each corresponding orientation group, generate new orientation vectors based on the obtained position control points and given initial orientation vectors (see Section 4.2). Based on the decision made for the position curve, use a modified version of the respective algorithm that was applied to the position group to get the solution for the orientation group (see Section 6).

4. If we obtain desired results, we continue with the next position group.

5. In case a solution could not be found, we need to detect where the algorithm failed in the orientation group and decide where we need to insert additional control points to form a constellation that can be solved. Insert respective additional knots in the knot vector and recalculate position control points according to new knots (see Section 6).

6. For the orientation group we regenerate orientation vectors according to changed position control points and resolve.

7. If all groups have been processed, connect all groups to double NURBS curve (see Section 7).

4 Allocating Groups

4.1 Splitting into Position Groups

As a first step, we need to partition the given sequence of position points into small curve segments. The curve segments are represented as groups (or sequences) of consecutive input points. We want to distinguish between simple and complicated groups. As handling complicated groups will be done in a more time-consuming processing step, it is desirable to keep the number of complicated groups as small as possible and to keep the complicated groups themselves as small as possible. A complex group is a group that contains at least one sharp corner, i.e., an angle smaller than a certain threshold α_{sharp}. Consequently, simple groups are those with no sharp corners. For complicated regions of the path we apply the idea of threading splines through 3D channels (see Section 5.1), for simple ones we use the idea of local non-rational cubic approximation (see Section 5.2).

We implement different grouping criteria based on the distances between consecutive input points and the incident angles between the two edges connecting three consecutive input points, see [SL11].

First, we deploy some global splitting criteria. As processing time for groups increases superlinearly with increasing group size, groups shall not exceed a certain upper limit of points n_{max}. Also, very long distances between consecutive points may make it difficult to optimize for curvature. Hence, if two consecutive input points exhibit a distance larger than a certain threshold d_{max}, the two points shall belong to two different groups.

To make the simple groups as large as possible, we proceed as shown in Figure 5. We iterate through the points of the input curve. When a new group contains more than five points with no sharp corner, we generate a new group that is marked as simple. We keep on growing that group until the maximum number of points n_{max} is reached or we are approaching a sharp corner. To detect an approaching sharp corner, we use a look-ahead method. We refer to n_{ahead} as the number of points used for the look-ahead. If this look ahead reports a sharp corner, we finalize the simple group and start a complicated group. If a sharp corner is reported before the group reaches the fifth point, the group is marked as complicated and we keep on adding points. A complicated group is finalized when n_{ahead} points have been added since the last sharp corner and no new sharp corner has been reported by the look ahead. When we have a sequence of more than n_{max} points with many angles greater than α_{sharp}, there is no possibility to finalize the current group without violating the n_{max} restriction. The solution is to check angles of the last $2 \cdot n_{ahead}$-1 points, and finalize the group with the point which is situated in the middle of the two points with the sharpest corners. It prevents the group to stop at the initial point with the sharpest corner.

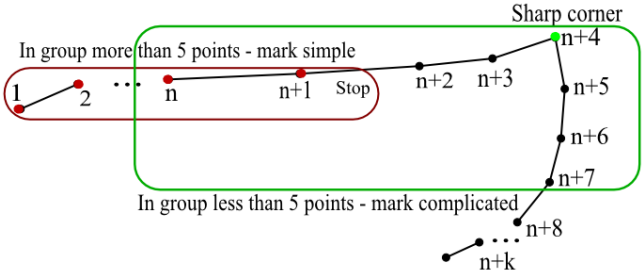

Figure 5: Complicated and simple regions.

The described solution produces complicated groups of minimum size. Moreover, the solution for a complicated group with a single sharp corner is symmetric in the sense that the sharp corner is at the middle point of the group, see Figure 6. Such a symmetric solution is desirable for milling applications, as it ensures that the milling result to both sides of the sharp corner is similar.

Such a symmetric behavior is also desirable when dealing with a group of multiple sharp corners. In particular, a common geometric feature of milling paths is that of a turn, i.e., a small sequence of points where the direction of milling in changed to its inverse direction at some offset. We handle such turns explicitly to

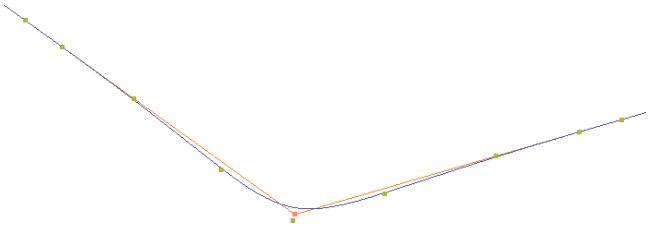

Figure 6: Symmetric solution for a single corner point, where the red curve is the control polygon and the blue curve is the solution.

assure that the middle of the turn corresponds to the middle of the generated group.

Depending on which criterion caused the formation of a group, we have different methods of how a group is to be connected with the subsequent group when generating the overall curve. We distinguish between two types of connections, namely a *line connection* and an *overlapping connection*. As mentioned above, if we have a pair of neighboring points a_k and a_{k+1} in distance more than d_{max}, we are finalizing the current group with the point a_k and start the next group with point a_{k+1}. This type of connection is what we call a line connection. Examples for line connections are shown in Figure 7.

To fulfill continuity requirements, we insert three additional points lying equidistantly on the line $a_k a_{k+1}$. We adjust the two groups by adding the three additional points to the end of the first group (i.e., after a_k) and to the beginning of the second group (i.e., before a_{k+1}). When optimizing the curve segments as described in the subsequent sections, we restrict the solutions to those that do not change the inserted points. Consequently, we assure that we maintain G^2-continuity across the line connection.

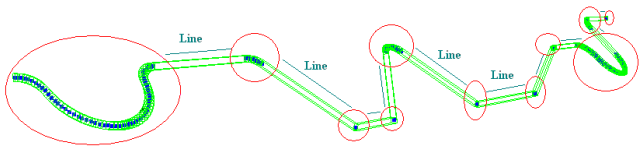

Figure 7: Line connections between groups of points that are further apart than distance d_{max}. Dots represent control points, which form groups encircled in red that are connected by straight lines.

Note that individual points can form a group, if they are more than distance d_{max} away from both its preceding and succeeding point. In this case that single point is connected with two line segments in a symmetric fashion as shown in Figure 6.

If two successive groups have not been separated by the maximum-distance criterion, the two groups are close together and, in general, have not been split in an area of low curvature. Figure 8 shows such a critical area, where the two groups come together at an area of rather large curvature. In this case, it is likely that the optimization of the first group generates a solution with the endpoint on the border of the tolerance channel. This fact will make it hard to produce a G^2-continuous transition with low curvature terms between the two groups. Thus, we want the two groups to overlap, i.e., we want the groups to share a small area. Consequently, we call this connection an overlapping connection. The overlap is handled as follows: we first get a solution for the first group and then we insert at the beginning of the second group the last few points from the first group. Now the starting point of the second group, in general, lies well within the tolerance channel making it easier to construct a good solution for the second group.

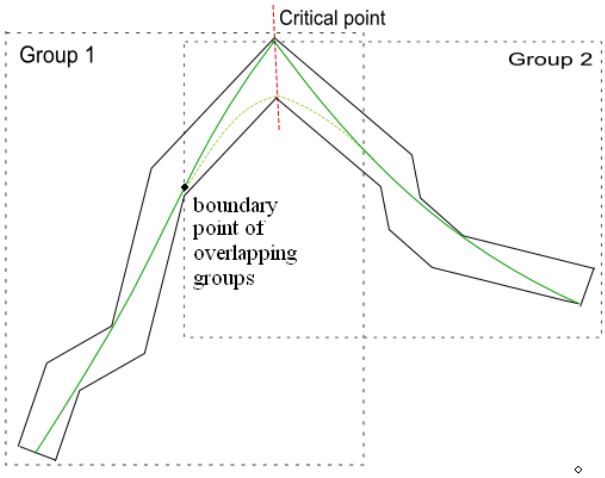

Figure 8: Necessity for overlapping connection at critical area.

The generated groups are sufficiently small to efficiently generate optimized solutions using the algorithms described in Sections 5.1 and 5.2. However, in some cases the algorithms cannot find a solution. If that happens, additional points are added to the initial sequence and the algorithms start over again with the refined point sequence. In our experiments, it turned out that this successive refinement and optimization step can be rather computationally intense. Instead, we

empirically found a criterion that estimates whether this refinement step is likely to happen. If yes, we insert points beforehand, which can significantly reduce the computation costs. Using the notations of Figure 9, the criterion can be formulated as follows: We insert the midpoint between B and C, if the angle α at B is small and the distance between B and C is large when compared to the tolerance ε of the channel. Those two criteria are combined such that the midpoint is inserted, if the point $(\cos\alpha, \|BC\|/\varepsilon)$ lies above the graph of the empirically found curve shown in Figure 9.

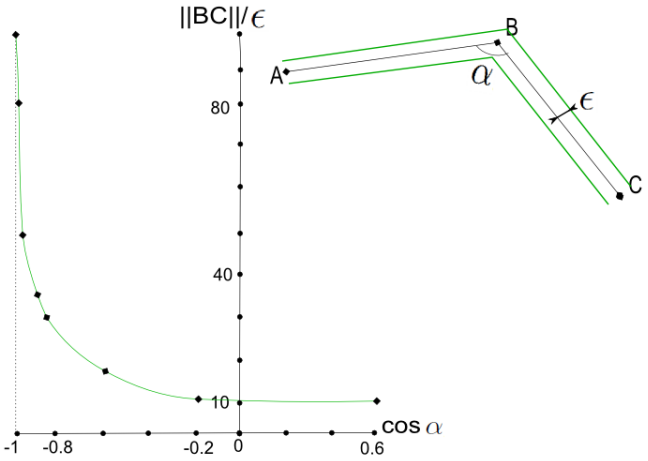

Figure 9: Point insertion criteria. If the distance divided by the tolerance of the channel is above the curve, then we insert the point in the middle between initial points B and C.

4.2 Adjusting Orientation Groups

To solve the orientation part, we first need to create new initial orientation points based on the already optimized position curve and given initial data. The location of the new orientation points has to be calculated in such a way that these points and the corresponding position control points form new orientation vectors. Direction of the vectors is considered as the combination of the directions of the two closest initial vectors proportional to the distance between the optimized position control point and the initial position points which form these closest vectors. Given a position control point, we detect to which interval of initial position points the position control point belongs. We distinguish three different cases, which are depicted in Figure 10. For each control point P_i we compute lines that go through the control point P_i and are perpendicular to the lines of the initial position curve $B_0, \ldots, B_m$. We compute those perpendicular lines for the closest two initial position curve segments $B_{i-1}B_i$ and B_iB_{i+1}. We determine whether the intersection of the perpendicular lines with the initial position curve segments belongs to the initial position curve. The following cases occur (cf. Figure 10):

- Point P_1: one of the perpendiculars at point P_1 intersects with line B_0B_1 inside the interval (solid black line) and another one intersects with outside part of line B_1B_2 (dashed black line). Hence, point P_1 lies in the first interval B_0B_1 and the closest initial orientation vectors are V_1, V_2.

- Point P_3: two perpendiculars at point P_3 intersect with the lines B_1B_2 and B_2B_3 not inside the intervals. Then, we compute the distances of the intersection point to the control point P_3 (blue lines) and P_3 relates to that segment with shorter distance.

- Point P_4: two perpendiculars at point P_4 intersect with lines B_2B_3 and B_3B_4 inside the intervals. Then, we again make the decision based on the distance as in the preceding case.

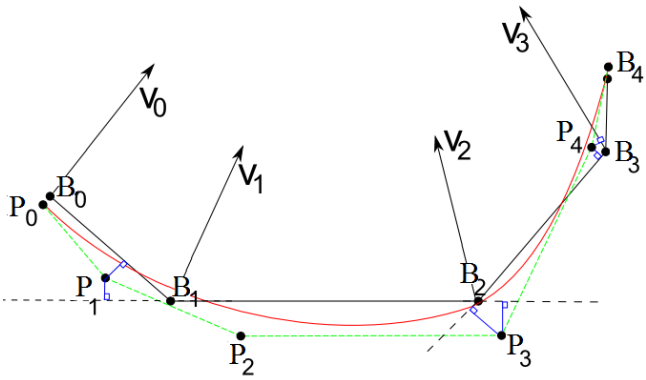

Figure 10: Control points determination to a proper interval. Arrows $V_0, \ldots, V_4$ indicate initial orientation vectors, where the starting points of these vectors are the initial position points $B_0, \ldots, B_4$. $P_0, \ldots, P_4$ are optimized control points of the local position B-spline curve (red curve), blue lines are perpendiculars to lines formed by $B_0, \ldots, B_4$.

After we have found the relating initial position curve segment, we built a new orientation vector from the position control points with the direction linearly

interpolated between the closest initial vectors in proportion to the distance from them to the current position control point.

5 Position Curve Optimization

In this section we present the algorithm for optimizing the position curve. This is equivalent to the optimization for 3-axis machining. Consequently, this step is equivalent to the approach described previously [SL11]. For a comprehensive description of the entire approach, we briefly describe the main aspects of the algorithm. The main reason of splitting data into groups is the superlinear time complexity of the algorithm for complicated groups. We have two types of connections between groups: lines and overlapping. Separation of groups by line is applicable when the groups end at neighboring points that are located in a distance more than some threshold d_{max}. Between these groups we can construct a straight-line solution without spending any calculation time. Overlapping connections are necessary when we have a large number of points close to each other and collecting these points in one group might increase running time of the algorithm significantly, so we have to separate them. On the other hand, separating these points in different groups might occur in areas with high curvature and it is not possible to have feasible local solutions. In this case, we have overlapping connections by including the same points at the end of the first group and at the beginning of the second group. Then, we can find solutions locally and the overlap assures a proper transition. For details we refer to the literature [SL11].

5.1 Local Sleeve Approach for Complicated Group

The handling of complicated groups is done by executing a local method based on the slefe approach by Lutterkort and Peters [MP05, LP01, LP99]. Slefe is short for "subdividable linear efficient function enclosure". The idea is to generate two polygons that represent lower and upper boundaries of a given B-spline curve. The construction can be generalized to curves in space by applying the construction in both coordinate-axis directions of the underlying 2D domain, see Figure 11. It is, then, referred to as the sleeve approach. After construction of the piecewise linear sleeve, it is sufficient to constrain this sleeve rather than the original nonlinear curve to stay within the channel. Carefully formulated, this approximation results in a linear feasibility problem that is solvable using linear programming.

The construction of a sleeve around the spline curve is equivalent to enclosing the spline curve with linear pieces. However, in the channel problem, the coefficients of the spline curve are unknown and are sought as the solution of the feasibility problem.

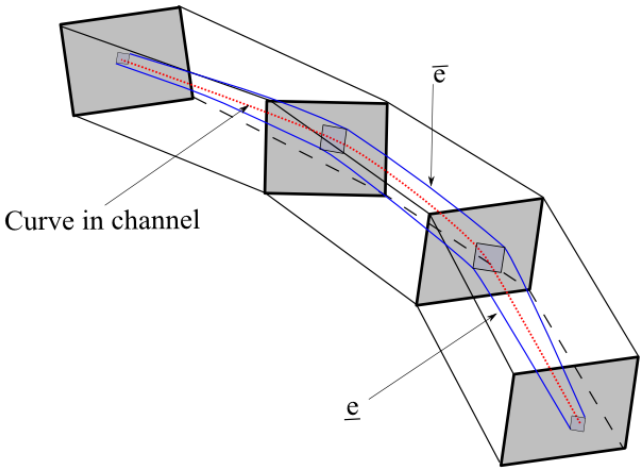

Figure 11: Sleeve $\bar{e}$, $\underline{e}$, and channel

Given a B-spline curve

$$C(u) = \sum_{j=0}^{m} P_j N_j^d(u),$$

with control points P_j. Let the B-spline basis functions N_j^d of degree d be defined over a knot vector (u_k). The control polygon $l(u)$ of the spline is the piecewise linear interpolant to the control points P_j at the Greville abscissa

$$u_j^* = \sum_{i=j+1}^{j+d} u_i/d,$$

i.e. $l(u_j) = P_j$.

The weighted second differences $\Delta_2 P_i$ of the control points are defined as

$$\Delta_2 P_i = P_{i+1}' - P_i', \qquad P_i' = \frac{P_i - P_{i-1}}{u_i^* - u_{i-1}^*}$$

In addition, we define $\Delta_i^- = \mathbf{min}\{0, \Delta_2 P_i\}$ and $\Delta_i^+ = \mathbf{max}\{0, \Delta_2 P_i\}$.

Over the interval $[u_k^*, u_{k+1}^*]$ the contribution of the i-th B-spline to the distance between a spline and its control polygon is captured by the non-negative and convex functions

$$\beta_{ki}(u_k^*) := \begin{cases} \sum_{j=i}^{\bar{k}}(u_j^* - u_i^*)N_j^d, & i > k, \\ \sum_{j=\underline{k}}^{i}(u_i^* - u_j^*)N_j^d, & i \le k, \end{cases} \quad (3)$$

(4)

where $\bar{k}$ and $\underline{k}$ are the indices of the first and last (at most) $d+1$ B-spline basis functions N_j^d whose support spans u_k^*.

Then, we can formulate the restrictions for our curve by

$$\underline{e}(u) \le C(u) \le \bar{e}(u)$$

where

$$\bar{e}(u) = l(u) + \mathcal{L}(\sum_i \Delta_i^+ \beta_{ki}(u_k^*), \sum_i \Delta_i^+ \beta_{k+1,i}(u_{k+1}^*)),$$

$$\underline{e}(u) = l(u) + \mathcal{L}(\sum_i \Delta_i^- \beta_{ki}(u_k^*), \sum_i \Delta_i^- \beta_{k+1,i}(u_{k+1}^*)),$$

with $u \in [u_k^*, u_{k+1}^*]$ and

$$\mathcal{L}(a_1, a_2) = a_1 \frac{u_{k+1}^* - u}{u_{k+1}^* - u_k^*} + a_2 \frac{u - u_k^*}{u_{k+1}^* - u_k^*}.$$

The constraints force the sleeve, and thus the spline, to stay inside the channel. We solve the linear programming problem using a simplex approach. The target function we intend to minimize is $\sum_{i=1}^{m-1}\sum_{j\in\{x,y,z\}}(\Delta_{i,j}^- + \Delta_{i,j}^+)$. By minimizing the absolute second differences we are minimizing the curvature of the spline. Sometimes the constraints cannot be met. In this case, we need to insert further points as described in Section 4.2.

5.2 Local G^2- continuous Spline Approximation for Simple Groups

Following the algorithm described in the approach presented by Piegl [PT97] and its modifications presented in previous work [SL11] we generate a local approximation scheme for simple groups using two cubic Bézier curves to assure G^2-continuous connections. Given a sequence of input points q_k,..., q_{k+n}, we set the first control point of the first curve and the last control point of the second curve to q_k and q_{k+n}, respectively. Using the notations of Figure 12, we set

$$P_0 = q_k, \quad R_3 = q_{k+n}$$

The endpoint interpolation property assures C^0-continuity. To assure G^1-continuity, we set

$$P_1 = P_0 + \alpha T_s \quad \text{and} \quad R_2 = R_3 + \beta T_e,$$

where T_s and T_e are the start and end unit tangents [PT97]. The unit tangents T_s and T_e as well as the values for α and β are defined by the continuity constraints with the preceding and succeeding group.

For G^2-continuity at the group's beginning we set

$$P_0 - 2P_1 + P_2 = A,$$

where A is determined by the preceding group (or A=0 for the first group, i.e., if k=0) [PT97]. Using the notations of Figure 12, the respective continuity requirements where the two Bézier curves stitch together are given by

$$\begin{aligned} P_3 &= R_0, \\ P_2 - P_3 &= R_0 - R_1, \\ P_1 - 2P_2 + P_3 &= R_0 - 2R_1 + R_2. \end{aligned}$$

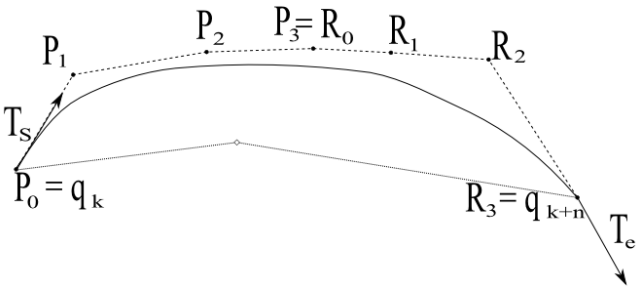

Figure 12: Local non-rational cubic curve approximation

To check for validity, we build the tolerance channel and the sleeve as described in Section 5.1. If the sleeve lies inside the tolerance channel, the approximation is sufficiently precise. If not, we do the steps described in Section 6 and then try to solve the problem for the modified sequence.

6 Orientation Curve Optimization

We want to generate a smooth B-spline curve for the orientation of the milling tool based on the obtained position curve and the newly generated orientation vectors. We can see in Figure 13 that the orientation curve repeats the sharp angles of the position curve, but we want to solve the orientation problem independently from the position curve. Therefore, we transform the problem of finding the orientation control points in space to the problem on the unit sphere. To

reformulate the problem, we start with orientation vectors which are given as the difference between position control points and respective interpolated orientation points. We can imagine that the orientation optimization problem is defined on a sphere by moving all position points to the sphere's center (see Figure 14). After that we solve the problem with the same approach that was used for the position curve with the modification that the tolerance channel is replaced by a tolerance frustum, see Figure 15. The frustum represents the tolerances for lead and tilt in the orientation vector. They can be specified independently. In the end we find the solution to the orientation curve by adding the computed orientation vectors to the corresponding position control points, i.e., by transforming back from the sphere to the spatial representation. We can do this operation without loss of continuity because by summing up corresponding control points of the different B-spline curves we get a new B-spline curve (see Figure 16).

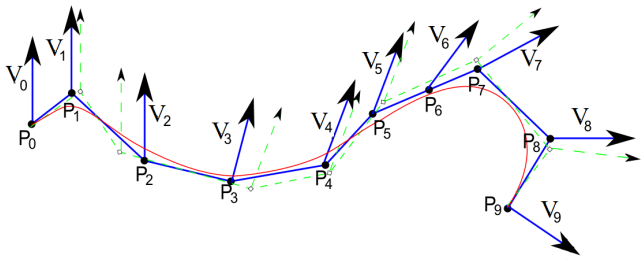

Figure 13: Green line connects initial position points, green dashed vectors are initial orientation vectors, red line is a B-spline position curve, blue line is the control polygon of B-spline position curve with control points $P_0, ..., P_9$, blue vectors $V_0, ..., V_9$ are generated new orientation vectors starting from position control points.

To apply the local sleeve approach for complicated groups and the spline approximation for simple groups for solving the orientation curve problem we have to modify a few parts. The main distinction from the position curve optimization algorithm is using the frustums to keep the control points inside them (see Figure 15). So, we need to add these constraints additionally to the other restriction we have.

In some cases, we might have an unfeasible orientation group problem. To solve this problem, we need to achieve greater flexibility of the desired B-spline curve by adding additional control points. We find a place in the orientation curve where the insertion of a new

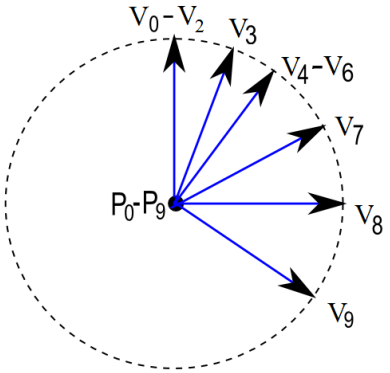

Figure 14: Described algorithms use the sphere representation to solve the orientation part of the problem. Generated orientation vectors $V_0, ..., V_9$ from the Fig. 13 are placed at the origin of a sphere such that, control points $P_0, ..., P_9$ coincide. We can see that V_0, V_1 and V_2 are the same for this example, because they have the same directions in Fig. 13.

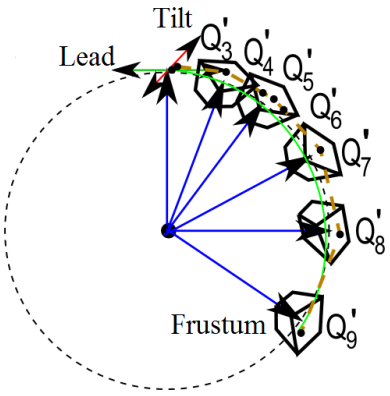

Figure 15: Frustums are built around every vector to restrict control points Q'_i to lie inside these frustums. Control Points Q'_4, Q'_5, Q'_6 lie inside matching frustums formed around collinear vectors V_4, V_5, V_6 but they do not need to be placed at the same position. Brown dashed line is a control polygon $Q'_0, ..., Q'_9$ of the green B-spline curve.

DOI 10.20385/1860-2037/15.2018.2, ISSN 1860-2037

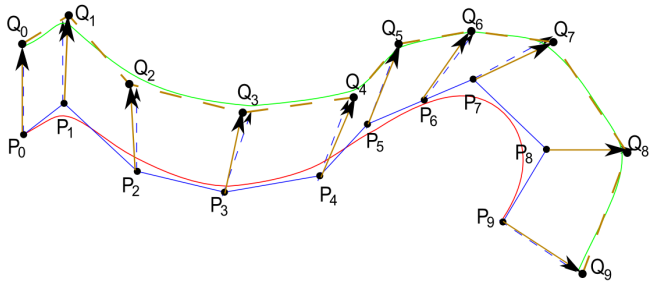

Figure 16: Obtained solution on the sphere is transformed back to the initial problem using formula $Q_i = P_i + Q'_i$. Brown vectors are the final orientation of the tool, red line is a position B-spline curve, green line is a orientation B-spline curve, dashed brown line is a control polygon of orientation curve. The two B-spline curves have the same knot vector.

additional control point is required. Usually the solution fails in places where we have a longer distance between some control points than in the rest of the group. Thus, we want to insert new control points in the middle of this distance. Then, we insert a respective knot in the existing knot vector and recalculate dependent control points for the position sequence of the control points, build new orientation vectors for the changed points, and resolve the orientation group. Thus, even if we have to insert additional control points in the orientation curve, we keep proper orientation vectors.

The procedure of the point insertion for position B-spline curve $\mathbf{C}(u)$ as in Equation (1) with a knot vector $\mathbf{U}$ as in Equation (2) is the following [PT97]:
Given a knot $\bar{u} \in [u_k, u_{k+1})$ that we want to insert into $\mathbf{U}$ to form the new knot vector $\mathbf{U}' = (u'_0 = u_0, \ldots, u'_k = u'_k, u'_{k+1} = \bar{u}, u'_{k+2} = u'_{k+1}, \ldots, u'_{m+1} = u_m)$, we recalculate the control points using Equation (5) by

$$\begin{aligned} P'_{k-d} &= P_{k-d}, \\ P'_i &= \alpha P_i + (1-\alpha) P_{i-1},\ k-d+1 \le i \le k, \\ P'_{k+1} &= P_k. \end{aligned}$$

For $i = k-d+1, \ldots, k$ we have

$$\alpha = \frac{\bar{u} - u_i}{u_{i+d} - u_i} \tag{5}$$

We need to generate a new control point in the middle of the interval $P_{i-1}P_i$ then build a new orientation vector from this point and frustum around it to get an orientation control point at the desired place. To achieve this, we set $\alpha = \frac{1}{2}$. Hence, from Equation (5) we obtain $u = \frac{u_{i+d}+u_i}{2}$. Afterwards, we recalculate points $P'_{i-d+1}, \ldots, P'_i$ and try to solve the orientation problem again.

7 Combining Groups

To complete the process, we need to put all groups together in two B-spline curves with common knot vector. We have to assure G^2-continuity at the connections. To fulfill G^2-continuity requirements in case of a line connection, we insert three additional points lying on the line $a_k a_{k+1}$, where neighboring initial points a_k and a_{k+1} belong to different groups. We adjust two groups by adding the three additional points to the end of the first group (i.e., after a_k) and to the beginning of the second group (i.e., before a_{k+1}).

For an overlapping connection of two groups, we need to distinguish whether the first group is a complicated or a simple one. If the first group is a complicated group, we calculate a new control point lying on the curve. We obtain this by double knot insertion (see [SL11]), cutting off the control points after this point, and adjusting the knot sequence by removing knots after the inserted ones. The second group will then start with the last control point of the first group. We did not modify the shape of the first group's curve. If the first group is a simple group, we delete the last few control points until we find a control point that was an input point. Since Bézier curves are endpoint interpolating, we can start the second group with that control point. When solving for the second group, we have to pick the first three control points of that group such that G^2-continuity is achieved. G^0-continuity requires that the last control point of the first group has to be equal to the first control point of the second group, for G^1-continuity we need to have the proportional tangent vector of the adjacent parts, and the G^2-continuity condition requires equality of the second derivatives of the adjacent parts. Using notations of Fig. 17, we obtain:

$$\begin{aligned} P_k &= R_1, \\ \frac{P_{k-1} - P_k}{||P_{k-1}, P_k||} &= \frac{R_1 - R_2}{||R_1, R_2||}, \\ P_{k-2} - 2P_{k-1} + P_k &= R_1 - 2R_2 + R_3. \end{aligned}$$

The procedure of combining groups is described in [SL11] in details for position curves in 3-axis matching and it is the same for the orientation curve.

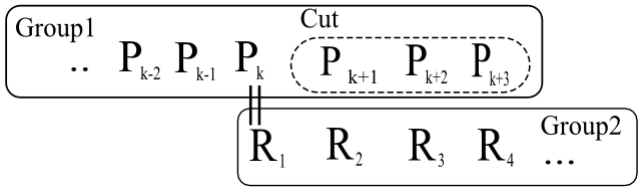

Figure 17: Connecting groups with G^2-continuity.

8 Results and Discussion

We first want to present results that investigate the correctness and quality of the position curves. These are results that are directly applicable to 3-axis machining, cf. [SL11]. Figure 18 shows a zoomed-in view of an area with sharp corners next to straight lines. We can observe that the curve always stays within the tolerance channel, while exhibiting a smooth behavior.

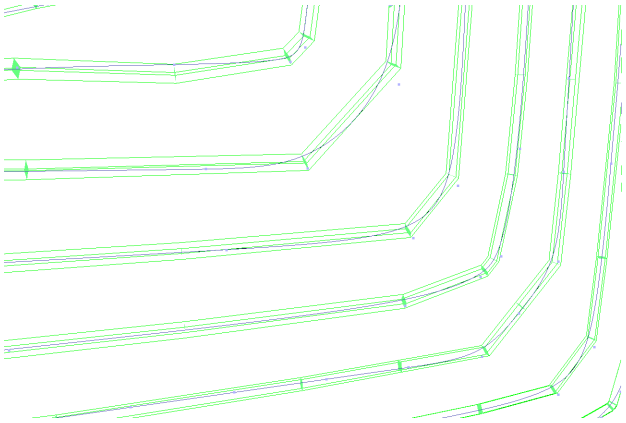

Figure 18: Smooth curve (blue) within tolerance channel (green) at sharp corners.

We compare the quality of the three local approaches with our grouping, i.e., local sleeve (left), local Bézier (right), and our combined approach (middle), in Figure 19. It can be observed that the local sleeve approach and our approach produce desirable results, where our approach is much faster, while the local Bézier approach is producing undesirable results, as the resulting curve exhibits some oscillating behavior, where the curve first bends too far to one side, which is then compensated by a bending to the other side. This is due to the fact that the Bézier curve only considers few points during the generation such that a globally smooth behavior is not well achievable.

Next, we want to consider both position and orientation curves as necessary for 5-axis machining. In Figure 20 we show a zoomed-in version of a real 5-axis tool path that was optimized with our approach. We can observe the correctness of the solution. Generated orientation vectors have similar directions as the initial ones and control points of the orientation curve stay inside the frustums.

Figure 21 depicts the individual steps of the optimization of the 5-axis tool path for a 90° turn. Using initial data of the orientation curve and control points of the position curve, we build frustums around the newly generated orientation vectors from position control points and get a solution with orientation control points inside the corresponding frustum.

Next, we want to investigate the computation time of our approach. For our experiments, we used a workpiece which mostly consists of almost straight parts, rounded parts, and turns as shown in Figure 22 with 209 input pairs of position and orientation points. Computation time for the 5-axis orientation algorithm is 0.735 seconds with the tolerance of the channel being 0.0008 mm and lead and tilt tolerance being 0.1°. The respective 3-axis optimization when only considering the position curve takes 0.38 seconds with the same tolerance of the channel. It takes less than double time to compute double amount of initial points because for almost straight areas of orientation parts on the sphere we have a very simple solution which we can get right away or we need to insert much less (or even none) additional points when compared to the position part.

We calculate the average and maximum curvature [SL11] of the orientation B-spline curve separately for different parts of the workpiece. Table 1 lists the computed curvature values. The upper row was calculated using tolerances of lead and tilt equal to 0.5°, the lower row using tolerances equal to 1.2°. We can observe that for line and simple parts the tolerances of lead and tilt do not influence the curvature because the solution is simple and control points are lying inside a smaller frustum anyway. However, for rounded parts and turns we have high values of curvature and the higher values are obtained for smaller angle tolerances, because the curve is forced to make sharper turns. These tests were performed on the workpiece shown in Fig. 23.

We also calculate the deviation of the resulting orientation vectors from the interpolated orientation vectors in terms of lead and tilt. We denote by vector D the forward direction of the lead, where D is in the forward direction of the curve, $-D$ is the backward direction of

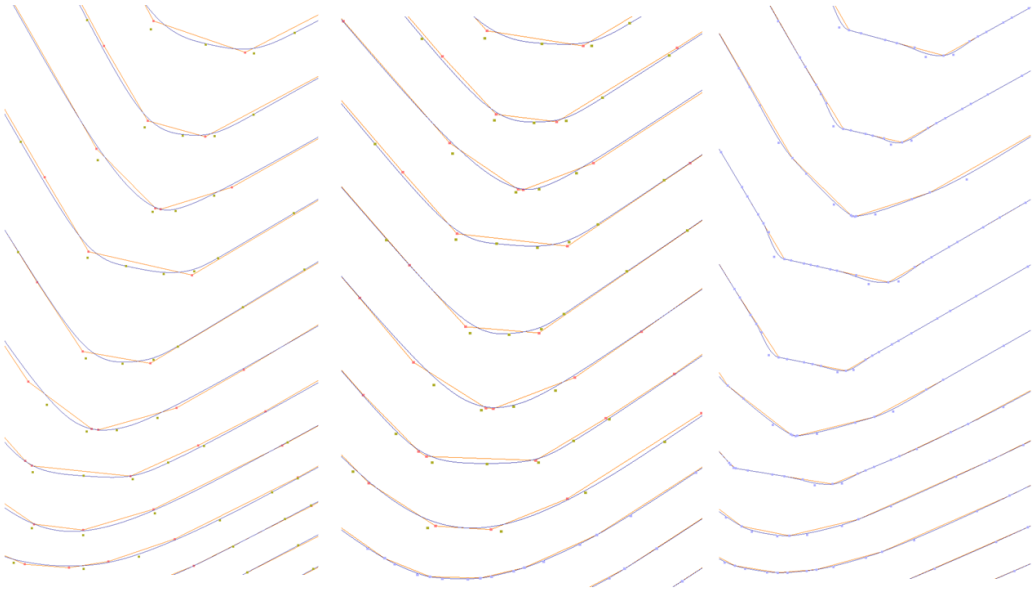

Figure 19: Results of local sleeve (left), local Bézier (right), and combined approach (middle) with our grouping.

	line parts		simple parts		rounded parts		parts with turn	
	av	max	av	max	av	max	av	max
1	1.8e-10	9.4e-10	0.0018	0.0377	1.111	2.2009	18.615	329.435
2	1.8e-10	9.4e-10	0.0018	0.0377	1.105	2.0455	18.3175	311.257

Table 1: Maximum and average curvature of parts of the workpiece parts separated in groups by type of initial data: line, simple parts with almost straight lines, rounded parts, and turns with small angles (see Figure 22). In the first test (upper row) we use tolerances of lead and tilt equal to 0.5°, in the second test (lower row) the tolerances are 1.2°.

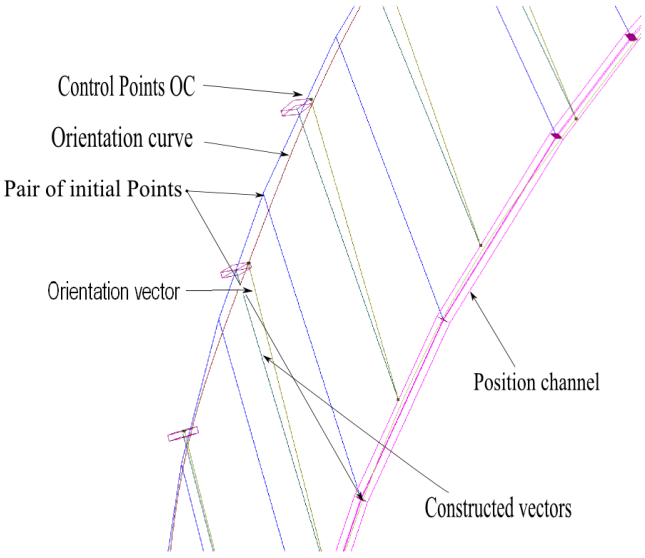

Figure 20: Optimized 5-axis tool path.

the lead. The vector R denotes the right direction of the tilt, where R is computed by the cross-product $D \times O$, where O is the orientation vector of the curve. Left direction of the tilt is the direction $-R$. For the lead, we compute the ratio of the angle between the resulting orientation vector and the plane that is formed by the left and right points of the frustum, and the given angle of lead tolerance. This ratio is 0 if the interpolated orientation vector and the resulting orientation vector are equal, and 1 if the maximum tolerance is used. The sign of the ratio indicates whether the orientation has been changed in forward or backward direction. Respectively for tilt, we compute the ratio of the angle between the resulting orientation vector and the plane that is formed by forward and backward points of the frustum, and the given angle of tilt tolerance

We can observe that for the part of the workpiece as in Figure 24 with allowed lead and tilt tolerance of 1° the values of deviation are not very high (see Figure 25). However, if we decrease the allowed tol-

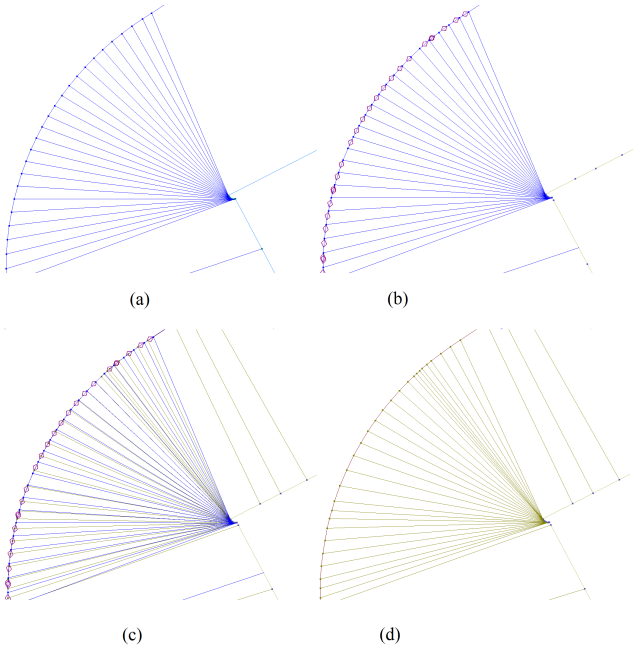

Figure 21: Outline of the process of generating a 5-axis tool path optimization for a 90° turn. (a) Initial position and orientation data. (b) Constructed frustums. (c) Generated orientation vectors and curve. (d) Resulting B-spline curves.

erance to 0.1° we can see that the deviation for some parts of the workpiece can reach to 100% in one direction and after that very quickly to the other direction (see Figure 27). This effect can be explained by the example shown in Figure 26. In Figure 26, we show the orientation vectors from Figure 24 on the sphere. The first few cyan orientation vectors and its frustums are almost identical. The subsequent vectors of different red colors are starting to slightly change the direction inside the same frustums. The reason is to keep the orientation curve straight. The resulting orientation vectors are located in one plane, but the positions are slightly changing in the direction of the subsequent frustum.

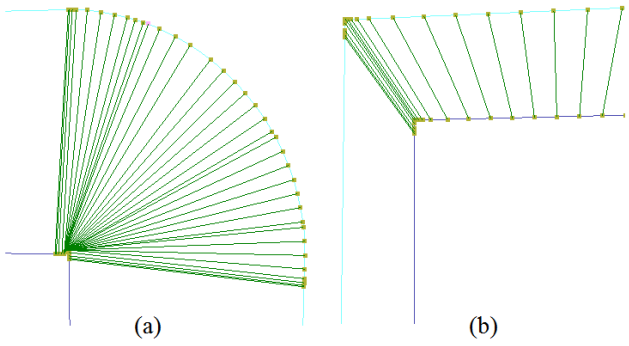

Figure 22: Parts of the workpiece. (a) Rounded parts (b) Turns and almost straight parts.

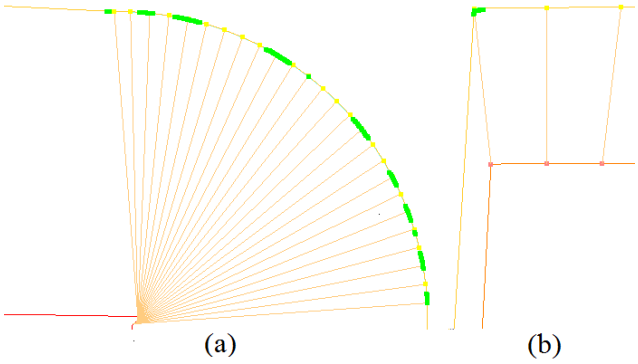

Figure 23: Parts of the workpiece. High curvature (higher than one) of the orientation curve are highlighted by green points, red line is a position B-spline curve, orange lines are orientation vectors and orientation polygon. (a) Rounded parts. (b) Turns.

On the workpiece with rounded parts and turns the deviation of the resulting vectors are shown in Figure 28. For the rounded part in Figure 28a, we can see a zoomed-in picture in Figure 29a, and for Figure 28b the view from the top is in Figure 29b. In Figure 29a, we can see that the orientation vectors are frequently changing its positions with respect to the generated vectors due to the minimization of the sum of the second differences. In Figure 29b, one can observe that the resulting deviation vectors before the turn changed the side with respect to the polygon formed by the centers of the frustums in order to decrease the curvature.

As parameters for our experiments, we define distance measures as multiples of the tolerance of the channel ε, which is half of the width of the channel. A good tradeoff in terms of computation time for solving individual groups and for splitting and joining overhead was found by setting the maximum number n_{max} of points per group to $n_{max} = 50$. A good value for the distance threshold d_{max} for splitting into separate groups was found to be $d_{max} = 300\varepsilon$. An angle is reported as a sharp corner, iff its cosine is smaller than $cos(\alpha_{sharp}) = -0.85$, because for corners that are sharper than this value the oscillation increases significantly when we apply the local spline approximation method.

Different splitting of the toolpath into groups may lead to different results of the overall curve. However,

Figure 24: Part of the workpiece. Yellow points are control points of the position curve, blue are orientation control points. Pink points are connection points between groups. Green lines are orientation vectors.

Figure 25: Diagram of lead (upper) and tilt (lower) deviation of the orientation vectors for a given tolerance of 1° for both lead and tilt and the example in Figure 24.

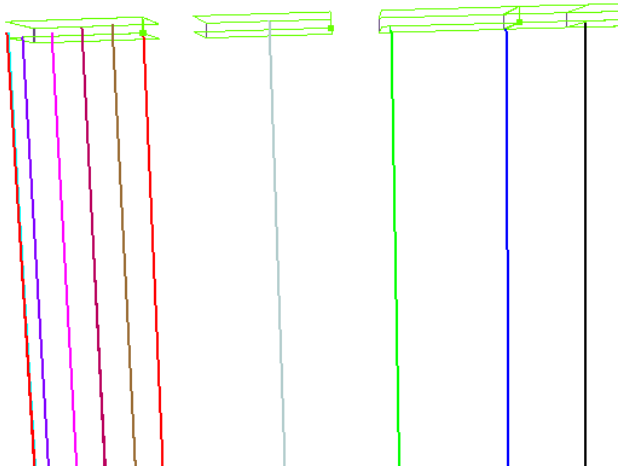

Figure 26: Resulting vectors and its frustums for the problem transformed into the sphere problem. All orientation vectors (lines of different color) start from one position, which is the center of the sphere.

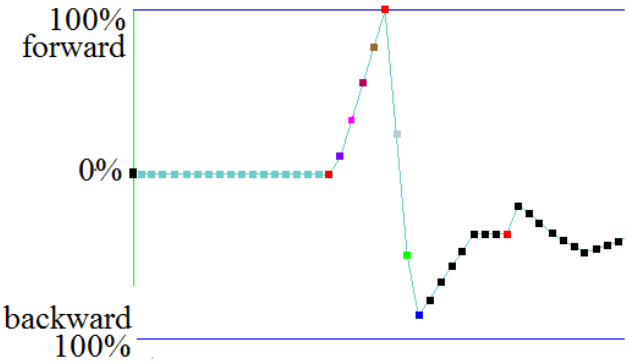

Figure 27: Diagram of lead deviation of the resulting vectors on the sphere for the example in Figure 24. The color of the points is the same as the color of the corresponding vector in Figure 26. Given tolerance is 0.1° for lead and tilt.

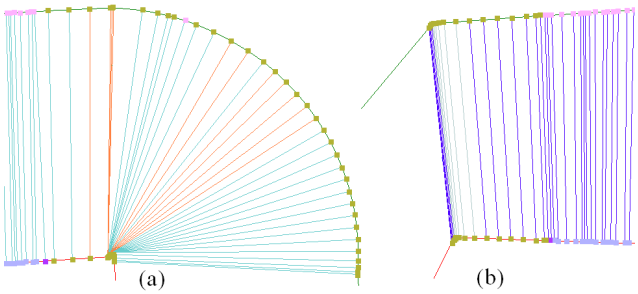

Figure 28: Parts of the workpiece. Points are control points, red curve is position B-spline curve, green curve is orientation B-spline curve. (a) Straight and rounded parts. Cyan and orange vectors are unit resulting orientation vectors, which indicate delay or overdrive of the resulting orientation vector in comparison to the initial one. (b) Turns. Dark purple and gray vectors are unit resulting orientation vectors. Different colors indicate different sides of the polygon formed by centers of the frustums.

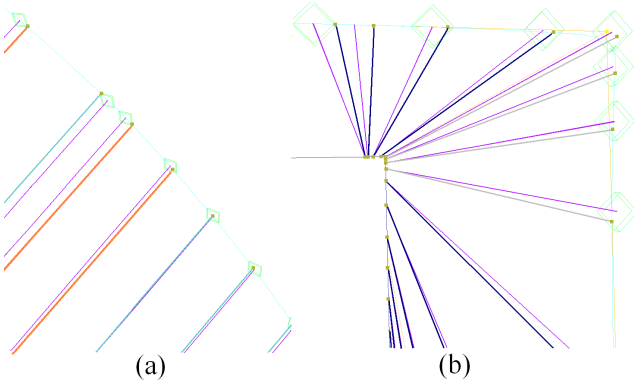

Figure 29: Parts of the workpiece. Purple lines are interpolated orientation vectors. (a) Zoomed-in rounded part, see Figure 28a. Orange lines are resulting vectors with delay, light blue are resulting vectors with overdrive. (b) View of the turn from the top, see Figure 28b. Gray and dark purple lines are resulting vectors on opposite sides of the polygon.

it is guaranteed that the curve remains within the band. Our splitting strategy is to use a strategy that is symmetric with respect to sharp positions, which leads to symmetric results in the case of symmetric geometry (as desired), and to make the simple groups as large as possible to get the highest reduction in computation time. Thus, our primary goal is to generate a desirable solution, and then the secondary goal is to make the generation as fast as possible.

DOI 10.20385/1860-2037/15.2018.2, ISSN 1860-2037

9 Conclusions

We presented an algorithm for generating curvature-optimized tool paths for 3-axis and 5-axis milling. The generated tool paths consist of a position B-spline curve in both cases (3-axis and 5-axis machining) and, in addition, an orientation B-spline curve in the case of 5-axis machining. The position curve can be handled like a 3-axis machining tool path. For the orientation curve, the control points are restricted to stay within a frustum formed by the lead and tilt tolerances.

To obtain an acceptable quality-speed ratio, we combine two algorithms: a local version of the sleeve-algorithm and a local spline approximation with G^2-continuity connections. Corresponding position and orientation point groups are solved by the same algorithm. We solve the orientation problem on the sphere to avoid any complications caused by location of position points.

The splitting of the original orientation curve into groups is performed with respect to the corresponding position point groups and complexity of the groups. The local spline approximation algorithm is very efficient and produces good results for simple groups, while the local sleeve algorithm is less efficient but produces high-quality results even for complicated groups.

The created algorithm can be used as a part of a micro-milling process. This approach provides new possibilities for calculating velocity, acceleration, and jerk profiles. The benefits are that profiles are easier to compute and that the milling can generally be performed with a higher velocity. The latter is essential for practical purposes, since the higher the velocity of the tool tip is the faster is the manufacturing process.

References

[FS01] Robert V. Fleisig and Allan D. Spence, *A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining*, Computer-Aided Design **33** (2001), no. 1, 1–15, ISSN 0010-4485, DOI 10.1016/S0010-4485(00)00049-X.

[JCL03] Cha-Soo Jun, Kyungduck Cha, and Yuan-Shin Lee, *Optimizing tool orientations for 5-axis machining by configuration-space search method*, Computer-Aided Design **35** (2003), no. 6, 549–566, ISSN 0010-4485, DOI 10.1016/S0010-4485(02)00077-5.

[KE02] Teajung Kim and Sanjay E.Sarma, *Toolpath generation along directions of maximum kinematic performance; a first cut at machine-optimal paths*, Computer-Aided Design **34** (2002), no. 6, 453–468, ISSN 0010-4485, DOI 10.1016/S0010-4485(01)00116-6.

[LDLB04] Jean Marie Langeron, Emmanuel Duc, Claire Lartiguec, and Pierre Bourdet, *A new format for 5-axis tool path computation, using Bspline curves*, Computer-Aided Design **36** (2004), no. 12, 1219–1229, ISSN 0010-4485, DOI 10.1016/j.cad.2003.12.002.

[LLL08] Sylvain Lavernhe, Christophe Lavernhe, and Claire Lartigue, *Kinematical performance prediction in multi-axis machining for process planning optimization*, The International Journal of Advanced Manufacturing Technology **37** (2008), no. 5, 534–544, ISSN 1433-3015, DOI 10.1007/s00170-007-1001-4.

[LLYM08] Wei Li, Yadong Liu, Kazuo Yamazaki, and Masahiko Mori, *The design of a NURBS pre-interpolator for five-axis machining*, The International Journal of Advanced Manufacturing Technology **36** (2008), no. 9, 927–935, ISSN 1433-3015, DOI 10.1007/s00170-006-0905-8.

[LP99] David Lutterkort and Jörg Peters, *Smooth Paths in a Polygonal Channel*, Proceedings of the Fifteenth Annual Symposium on Computational Geometry, SCG '99, ACM, 1999, pp. 316–321, ISBN 1-58113-068-6, DOI 10.1145/304893.304985.

Citation

Evgenia Selinger and Lars Linsen, *Efficient Error-bounded Curvature Optimization for Smooth Machining Paths*, Journal of Virtual Reality and Broadcasting, 15(2018), no. 2, February 2019, DOI 10.20385/1860-2037/15.2018.2, ISSN 1860-2037.

[LP01] David Lutterkort and Jörg Peters, *Tight linear envelopes for splines*, Numerische Mathematik **89** (2001), no. 4, 735–748, ISSN 0945-3245, DOI 10.1007/s002110100181.

[MP05] Ashish Myles and Jörg Peters, *Threading splines through 3D channels*, Computer-Aided Design **37** (2005), no. 2, 139–148, ISSN 0010-4485, DOI 10.1016/j.cad.2004.04.004.

[PT97] Les A. Piegl and Wayne Tiller, *The NURBS Book*, Springer, 1997, ISBN 3-540-61545-8.

[SL11] Jevgenija Selinger and Lars Linsen, *Efficient Curvature-optimized G2-continuous Path Generation with Guaranteed Error Bound for 3-axis Machining*, Proceedings of the 15th International Conference on Information Visualisation, IEEE, 2011, pp. 519–527, DOI 10.1109/IV.2011.31.

[SL18] Jevgenija Selinger and Lars Linsen, *Efficient Curvature-optimized G2-continuous Path Generation with Guaranteed Error Bound for 5-axis Machining*, Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018), vol. 1, 2018, pp. 59–70, DOI 10.1109/IV.2011.31.

[WMCH07] Yongzhang Wang, Xiongbo Ma, Liangji Chen, and Zhenyu Han, *Realization Methodology of a 5-axis Spline Interpolator in an Open CNC System*, Chinese Journal of Aeronautics **20** (2007), no. 4, 362–369, ISSN 1000-9361, DOI 10.1016/S1000-9361(07)60056-9.