Spline-based trajectory generation for CNC machines

Tim Mercy, Nicolas Jacquod, Raoul Herzog, and Goele Pipeleers

***Abstract*—Manufacturing of workpieces with CNC machines requires computing machine tool trajectories that fast and accurately track the desired workpiece contour. This paper presents a novel B-spline trajectory generation method for machine tools. The method solves an optimal control problem to minimize the motion time of the tool, while taking into account the velocity, acceleration and jerk limits of the tool axes. Furthermore, it directly includes the allowed workpiece tolerance, by constraining the trajectory to lie inside a tube around the nominal geometry contour. This allows exploring the trade-off between accuracy and productivity, while computing near-optimal trajectories. The presented method creates fluent connections between segments that build up the contour by simultaneously optimizing trajectories for multiple segments. On the other hand, limiting the amount of simultaneously optimized segments and using an efficient problem formulation keeps the computation time acceptable. The trajectory generation method is validated in simulation by comparison with industrial benchmarks, showing a reduction in machining time of more than 10%. The comparison to a state-of-the-art corner smoothing approach shows that the presented method obtains similar or slightly faster trajectories, at a computation time that is up to 45 times lower. In addition, the method is validated experimentally on a 3-axis micro-milling machine. To easily generate trajectories for different workpieces and machines, the method is included in a user-friendly open-source software toolbox.**

***Index Terms*—CNC machine tools, optimal control, splines, trajectory generation**

I. Introduction

MODERN manufacturing industry heavily relies on Computerized Numerical Control (CNC) machines for the production of various kinds of workpieces. Applications contain laser cutting of sheet metal parts, milling and lathing of machine parts, and water jet cutting of marble.

Manuscript received May 14, 2018; revised July 29, 2018 and September 18, 2018; accepted September 26, 2018. This work benefits from KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC), from the project G0C4515N of the Research Foundation-Flanders (FWO-Flanders), from the Flanders Make ICON project: Avoidance of collisions and obstacles in narrow lanes and from the KU Leuven Research project C14/15/067: B-spline based certificates of positivity with applications in engineering. Flanders Make is the Flemish strategic research centre for the manufacturing industry.

Tim Mercy and Goele Pipeleers are with MECO Research Team, Department of Mechanical Engineering, KU Leuven, BE-3001 Leuven, Belgium, and are members of DMMS lab, Flanders Make, BE-3001 Leuven, Belgium. (e-mail: tim.mercy@kuleuven.be).

Raoul Herzog and Nicolas Jacquod are with the University of Applied Sciences Western Switzerland, Yverdon-les-Bains, 1401, Switzerland.

Workpieces are typically created in a Computer Aided Design (CAD) program, and converted to a GCode file by a Computer Aided Manufacturing (CAM) system. This file represents the contour of the workpiece as a set of G-commands, using for example straight lines (G01), circle arcs (G02 and G03) or more complicated shapes, represented as splines (G05). In order to machine the workpiece, the machine tool needs to follow this contour accurately. To this end, the contour needs to be converted into trajectories for the different machine axes. These trajectories form a time domain representation of the contour, describing the position for all axes at every time instant.

When machining a workpiece, it is generally desired to maximize the productivity. Hence, the goal is to create the piece as fast as possible, while taking into account machine limits like the maximum velocity, acceleration and jerk, and process limits like the maximum allowed feed rate. In addition, there are also quality requirements, which translate into a certain tolerance on the workpiece. This tolerance is equal to the maximum contouring error, and creates a tube around the given contour, inside which the tool has to stay. The problem formulation above naturally translates into an Optimal Control Problem (OCP), which is challenging to solve, since it is non-convex and has a large time horizon.

In general, there are various approaches to solve OCPs, including dynamic programming, indirect methods which translate the problem into a two point boundary value problem, and direct methods which transform the OCP into a finite dimensional nonlinear programming problem. Direct approaches such as single shooting and multiple shooting, e.g. implemented in [1], or pseudospectral methods, e.g. implemented in [2], are a popular choice to formulate the OCPs at hand [3]. However, an important drawback of these methods is that they handle constraints that need to hold over the complete time horizon by using time gridding. In this approach, the time domain is discretized and constraints are only imposed at specific points in time. As a consequence, constraint violations may occur between sample points, such that a very fine grid is required to reduce the amount of violations. However, a finer grid leads to more constraints, and therefore a more complicated problem.

On the other hand, there are also tailored methods to handle CNC trajectory generation OCPs. Methods that solve these challenging OCPs in a single step are called coupled approaches. However, most methods are decoupled and solve the problem approximately [4], separating contour re-shaping from feed rate generation. Within the decoupled approaches

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

there are two classes. The first class purely focuses on feed rate planning and does not change the provided contour. Feed rate planning methods are often based on optimization, and include the machine and process limits as constraints. When only constraints up to axis acceleration are taken into account, the trajectory generation problem for following a path exactly can be written in a convex form, allowing to find the global optimum [5]. However, a discontinuity in the axis acceleration can lead to vibrations in the machine tool [6]. To avoid vibrations, the authors of [7] formulate a jerk-constrained nonlinear optimization problem. The downside of these methods is that they require following the contour exactly. This means that complete standstill is necessary at corners, which leads to high machining times. The second class of decoupled approaches solves this issue by first re-shaping the contour, such that it can afterwards be followed at a higher feed rate. The authors of [8] propose a discrete geometry optimization using B-splines, written as a quadratic program. Afterwards, a feed rate planning method, using e.g. a particle swarm optimization approach [9], a discrete minimum time OCP formulation [6], or an analytic approach [10], can convert the re-shaped path into trajectories.

Because decoupled approaches do not treat the complete problem in one step, they do, in general, not lead to a time-optimal movement [8]. Coupled approaches, on the other hand, combine contour re-shaping and feed rate planning into a single problem, allowing the computation of an optimal solution. These methods use the provided contour only as a guideline. The approach of [11] rounds the corners in the original contour and assigns a feed rate for every corner, using an iterative procedure to ensure that the required tolerance is met. In [12] the authors fit a minimum jerk spline through the provided contour. The method aims at minimizing the tracking error and it is not possible to constrain it to a specific value. The authors of [13] propose a nonlinear model predictive path following method that uses a spatial reformulation to obtain simple geometric constraints. While this method allows explicitly constraining the tracking error, the corresponding optimization problem is strongly non-convex and very hard to solve. In [14], a coupled path generation method is presented that obtains a very low solving time by making a convex approximation of the optimization problem at hand, and using down-sampling to reduce the dimension of the resulting quadratic program. By solving the problem online, with a receding horizon, uncertainties can be taken into account with a state observer. However, specific tuning is required to reach a solving time that makes the method real-time applicable. In [15], the authors propose a corner-smoothing approach that finds the trajectory that smoothly traverses a corner in a time optimal way, while taking into account the required tolerance and the kinematic constraints. The solution is obtained by transforming the problem into a couple of two point boundary value problems. However, the method can only handle corners, and is not suited to compute trajectories for circle segments. In addition, a certain minimum length of the segments that build up the corner is required, and the solution times to smooth a corner are high, ranging from 0.5 to $1.4\,\mathrm{s}$. Finally, in [16], the authors solve an OCP that includes an explicit tolerance constraint on the workpiece, allowing a significant reduction in machining time. However, the downside is that the tool center position is forced to move through the segment connection points, reducing the optimality of the trajectory.

In order to mitigate the drawbacks mentioned above, the current paper presents a novel B-spline based trajectory generation method for CNC machines. It is a coupled direct approach that parameterizes the tool trajectory as a B-spline and exploits spline properties to formulate the trajectory generation problem as a small-scale optimization problem, allowing a low solving time. In addition, these spline properties allow obtaining guaranteed constraint satisfaction at all time instants and avoid using time gridding. Limits on the axes velocity, acceleration and jerk are easily imposed. In addition, the tracking error is explicitly constrained in the optimization problem. This requires non-convex state constraints, but leaves the trajectory freedom to move inside a tube around the nominal contour. This freedom allows reducing the machining time, while the desired quality of the workpiece is still guaranteed. To obtain efficient contouring constraints, a separate B-spline is associated to each GCode segment that builds up the contour. Fluent transitions between subsequent splines are obtained by adding continuity conditions in their connection points. However, in order to keep the resulting OCP tractable, not all segments are optimized at once. Using a moving horizon approach, only the first few segments are optimized simultaneously. Afterwards, the trajectories for the first segment are saved, and the window is shifted forward one segment. In combination with an efficient problem formulation, the moving horizon approach leads to OCPs that have an acceptable solving time. Finally, as opposed to [16], it is not required that the trajectory passes exactly through the connection points of subsequent segments, allowing an increase in optimality.

The presented method is compared to an industrial trajectory generator from Siemens [17], and shows a reduction in machining time of 11% for a simple benchmark and 38% for a benchmark consisting of plenty of short segments. In addition, a comparison to the state-of-the-art method presented in [15] shows that the presented method computes similar or slightly faster trajectories, while the required computation times are up to 45 times lower. To prove its practical applicability, the method is experimentally validated on a 3-axis micro-milling machine. To facilitate generating trajectories for various workpieces, the presented method is embodied in OMG-tools, a user-friendly open-source Python toolbox [18].

Section II describes the general OCP and the moving horizon approach. Section III explains the spline-based trajectory generation in detail and shows how to keep the optimization problems small-scale by using a spline parameterization. Afterwards, Section IV and V validate the presented method by showing numerical simulation examples and an experimental validation. Finally, Section VI concludes the paper.

II. Problem formulation

The presented trajectory generation method aims at computing the machine tool trajectories that trace the provided contour as fast as possible, while satisfying the velocity,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

acceleration and jerk limits of the tool axes, and staying inside the desired tolerance band around the workpiece contour. This problem formulation naturally translates into an OCP, that is described more in detail below.

For the sake of clarity, the paper only discusses contours consisting of straight segments (G01) and circle arcs (G02 and G03), but the designed method also allows trajectory generation for spline contour segments (G05). For the moment, the method is only explained and implemented in 2D, while it is conceptually expandable to 3D. The considered GCode describes the workpiece contour for the tool center point. In the case of operations such as milling, where the machine tool has a certain diameter, this may require a preprocessing step to convert the provided workpiece contour to a GCode path for the tool center point.

First, Section II-A formulates the OCP that computes the trajectory for a single segment. Afterwards, Section II-B explains how to improve this formulation and set up an OCP that simultaneously optimizes trajectories for multiple segments.

A. Trajectory generation for a single segment

When formulating the trajectory generation problem for a single segment, the motion trajectories for the tool center point are given by:

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \tag{1}$$

in which $x(t)$ and $y(t)$ describe the axes positions as a function of time. The axes velocities, accelerations and jerks are given by the corresponding derivatives of $q(t)$.

The computed trajectory must steer the tool from the start of the segment q^{o} to the end q^{g}. In addition, the tool starts from, and ends in standstill, requiring the following boundary conditions:

$$\begin{aligned} q(0) &= q^{\mathrm{o}} \quad q(T) = q^{\mathrm{g}}, \\ \dot{q}(0) &= 0 \quad \dot{q}(T) = 0, \\ \ddot{q}(0) &= 0 \quad \ddot{q}(T) = 0, \end{aligned} \tag{2}$$

in which T represents the total motion time that is required to machine the segment.

Furthermore, the axes velocities, accelerations and jerks are constrained within their allowed bounds, at all time instants:

$$\begin{aligned} v_{\min} &\leq \dot{q}(t) &&\leq v_{\max}, \quad \forall t \in [0, T], \\ a_{\min} &\leq \ddot{q}(t) &&\leq a_{\max}, \quad \forall t \in [0, T], \\ j_{\min} &\leq \dddot{q}(t) &&\leq j_{\max}, \quad \forall t \in [0, T]. \end{aligned} \tag{3}$$

Hence, sufficiently smooth motion trajectories are required. In this formulation, the bounds are vectors, such that different limit values are allowed for both axes within $q(t)$.

The machine dynamics are not explicitly accounted for in the problem formulation. However, the kinematic constraints (3) will have a large influence on the dynamic behavior of the machine, e.g. tightening the jerk constraints will lower the vibrations in the machine [6]. Therefore, constraints (3) can be seen as kinodynamic constraints.

During machining, the allowed feed rate is typically constrained to $f_{\max}$, in order to obtain sufficient surface quality of the workpiece:

$$\|\dot{q}(t)\|_2^2 \leq f_{\max}^2 \quad \forall t \in [0, T]. \tag{4}$$

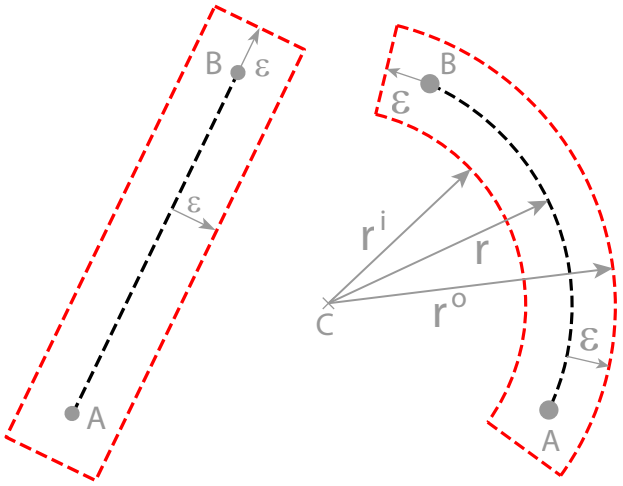


Fig. 1: Tolerances for a straight segment and a circle segment

In addition, contouring tolerance constraints are required to reach the desired accuracy. When formulated for the complete workpiece at once, these constraints become very complicated. However, they are greatly simplified when considering only a single GCode segment. For a straight segment, connecting A and B, adding a tolerance turns it into a rectangle, as shown in Figure 1a. The following constraint expresses that the perpendicular distance from the line through A and B to the tool center has to be smaller than the desired tolerance at all time instants:

$$-\epsilon \leq a^{\top} \cdot q(t) - b \leq \epsilon \quad \forall t \in [0, T], \tag{5}$$

in which ϵ is the provided contouring tolerance, a is the normalized normal vector of the segment, and b is the signed distance between the segment and the origin. On the other hand, adding a tolerance to a circle arc turns it into a ring segment, as shown in Figure 1b. The following constraint expresses that the tool has to stay inside the ring at all time instants:

$$(r - \epsilon)^2 \leq \|q(t) - \mathrm{C}\|_2^2 \leq (r + \epsilon)^2 \quad \forall t \in [0, T], \tag{6}$$

in which C is the center of the ring and r is its nominal radius. Note that constraints (5) and (6) consider the *infinite* versions of the segment. Equation (5) forces the tool to stay within a distance ϵ from the infinitely long line through A and B, instead of inside the rectangle shown in Figure 1a. On the other hand, (6) keeps the tool inside the complete ring, instead of inside the ring segment. Section II-B explains how to deal with the consequences of this approximation.

Finally, the goal is to obtain time-optimal trajectories. To be able to optimize the motion time T, the problem is reformulated using a dimensionless time $\tau = \frac{T}{t}$.

Assembling constraints (2)-(6) and using T as objective gives the OCP that computes time-optimal trajectories for a single segment. However, this formulation requires the tool to come to standstill at the end of each segment, in order to connect any subsequent segment in a smooth way. Only for exact path following (when ϵ is zero), these decoupled

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

trajectories form the optimal solution. When there is a nonzero tolerance, decoupling between segments is always suboptimal. Moreover, the suboptimality rises as the tolerance increases. The following section explains how to avoid decoupled trajectories.

B. Trajectory generation for multiple segments

By simultaneously optimizing the trajectories for a sequence of N segments, decoupling between subsequent segments can be avoided. In this approach, a separate trajectory $q_j(t)$ is assigned to each segment j, together with its own time horizon T_j. In addition, each segment gets its own contouring constraints, which therefore keep the simple form of (5) and (6). In this formulation, it is possible to specify a different tolerance for each segment in the contour.

In order to obtain a fluent connection between the trajectories that traverse subsequent segments, the following continuity constraints are added:

$$\begin{aligned} q_j(1) &= q_{j+1}(0), \\ \dot{q}_j(1) \cdot T_{j+1} &= \dot{q}_{j+1}(0) \cdot T_j, \\ \ddot{q}_j(1) \cdot T_{j+1}^2 &= \ddot{q}_{j+1}(0) \cdot T_j^2. \end{aligned} \tag{7}$$

These constraints express that the position, velocity and acceleration at the end of the trajectory associated with segment j have to be equal to the corresponding values at the beginning of the trajectory traversing segment $j+1$.

Assembling all elements from Section II-A, adapting them to account for multiple segments and adding (7), leads to the final OCP:

$$\begin{aligned} \min_{q_j(\cdot), T_j} \quad & \sum_{j=1}^{N} T_j \\ \text{s.t.} \quad & q_0(0) = q^{\mathrm{o}}, \quad q_N(1) = q^{\mathrm{g}}, \\ & \dot{q}_0(0) = 0, \quad \dot{q}_N(1) = 0, \\ & \ddot{q}_0(0) = 0, \quad \ddot{q}_N(1) = 0, \\ & v_{\min} \cdot T_j \le \dot{q}_j(\cdot) \le v_{\max} \cdot T_j \quad \forall j \in \{1, N\}, \\ & a_{\min} \cdot T_j^2 \le \ddot{q}_j(\cdot) \le a_{\max} \cdot T_j^2 \quad \forall j \in \{1, N\}, \\ & j_{\min} \cdot T_j^3 \le \dddot{q}_j(\cdot) \le j_{\max} \cdot T_j^3 \quad \forall j \in \{1, N\}, \\ & \|\dot{q}_j(\cdot)\|_2^2 \le f_{\max}^2 \cdot T_j^2 \quad \forall j \in \{1, N\}, \\ & -\epsilon \le a_j^\top \cdot q_j(\cdot) - b_j \le \epsilon \quad \forall j \in J_{\mathrm{G}_{01}}, \\ & (r_j - \epsilon)^2 \le \|q_j(\cdot) - \mathrm{C}_j\|_2^2 \quad \forall j \in J_{\mathrm{G}_{02},\mathrm{G}_{03}}, \\ & \|q_j(\cdot) - \mathrm{C}_j\|_2^2 \le (r_j + \epsilon)^2 \quad \forall j \in J_{\mathrm{G}_{02},\mathrm{G}_{03}}, \\ & q_j(1) = q_{j+1}(0) \quad \forall j \in \{1, N-1\}, \\ & \dot{q}_j(1) \cdot T_{j+1} = \dot{q}_{j+1}(0) \cdot T_j \quad \forall j \in \{1, N-1\}, \\ & \ddot{q}_j(1) \cdot T_{j+1}^2 = \ddot{q}_{j+1}(0) \cdot T_j^2 \quad \forall j \in \{1, N-1\}. \end{aligned} \tag{8}$$

In this formulation, $q(\cdot)$ is equivalent to $q(\tau), \forall \tau \in [0,1]$. $J_{\mathrm{G}_{01}}$ and $J_{\mathrm{G}_{02},\mathrm{G}_{03}}$ represent the set of straight and ring segments respectively, such that $J_{\mathrm{G}_{01}} \cup J_{\mathrm{G}_{02},\mathrm{G}_{03}} = \{1, N\}$.

When solving (8) for blocks of N segments, there is still a decoupling between segment N and $N+1$. In addition, the required standstill at the end of segment N influences the previous trajectories. To solve these issues, the OCP is inserted in a moving horizon approach. After computing the trajectories for the first N segments, merely the trajectory for the first segment is saved. Afterwards, the horizon is shifted: segment one is removed from the OCP, and segment $N+1$ is added. In the next step, the trajectories for segment two to $N+1$ are optimized simultaneously. Since standstill is only required at the end of the last segment in the horizon, while only the first trajectory is saved, decoupling is avoided: all obtained trajectories fluently connect subsequent segments. In addition, using the solution from the previous step as initial guess strongly reduces the solving time.

However, in general, the resulting optimization problem is still challenging to solve. The following section describes how to efficiently formulate this problem, allowing to swiftly obtain a solution.

III. Spline-based trajectory generation

This section first explains how to formulate problem (8) efficiently by using a spline parameterization for $q(\tau)$. Afterwards, it further details the implementation of the presented trajectory generation method.

A. Splines

In [19], the authors propose a method to transform OCPs of the form (8) into a small-scale nonlinear optimization problem. There are two key aspects of the method: (i) a B-spline parameterization is adopted for the motion trajectory $q(\tau)$; and (ii) the properties of B-splines are exploited to replace constraints over the entire time horizon by small, finite, yet conservative, sets of constraints.

Splines are piecewise polynomial functions, that can describe complex trajectories using only few variables. A spline is given by a linear combination of B-spline basis functions [20], as illustrated in Figure 2. Therefore, the motion trajectory is parameterized as:

$$q(\tau) = \sum_{i=1}^{n} c_i^q \cdot B_i^q(\tau). \tag{9}$$

Here $B_i^q(\tau)$ are B-spline basis functions, c_i^q are the spline coefficients and n follows from the spline degree and the number of knots. In order to obtain smooth acceleration trajectories and allow including jerk constraints, splines of degree three are used throughout this paper. When using this spline parameterization in (8), the variables of the problem become the coefficients c_i^q and motion time T. Since the motion trajectory $q(\tau)$ is a spline, the velocities $\dot{q}(\tau)$, accelerations $\ddot{q}(\tau)$ and jerks $\dddot{q}(\tau)$ are splines as well. Their coefficients $c_i^{\dot{q}}$, $c_i^{\ddot{q}}$ and $c_i^{\dddot{q}}$ are readily verified to depend linearly on c_i^q, e.g.:

$$\dot{q}(\tau) = \sum_{i=1}^{n-1} c_i^{\dot{q}}(c_i^q) \cdot B_i^{\dot{q}}(\tau). \tag{10}$$

The main reason for adopting the B-spline parameterization is the corresponding convex hull property, which states that a spline is always contained in the convex hull of its B-spline coefficients [20] (see Figure 2). This way, spline constraints can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

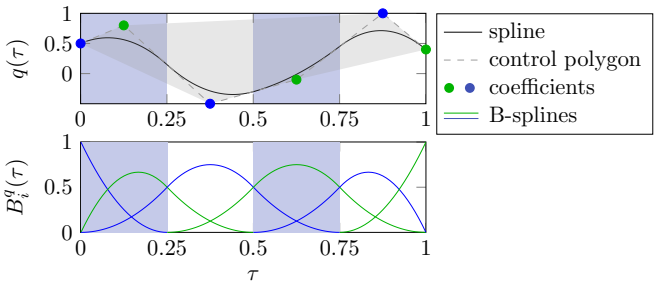


Fig. 2: A spline as a linear combination of B-spline basis functions

be enforced through constraints on the B-spline coefficients. For instance, the semi-infinite velocity constraints

$$v_{\min} \leq \dot{q}(\tau) \leq v_{\max}, \ \forall \ \tau \in [0, 1] \tag{11}$$

are guaranteed to hold if

$$v_{\min} \leq c_i^{\dot{q}} \leq v_{\max}, \ i = 1 \dots n. \tag{12}$$

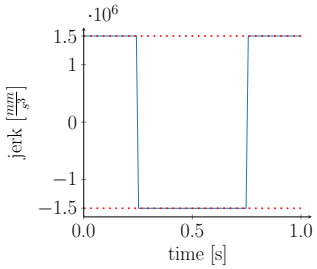
Replacing semi-infinite sets of constraints of the form (11), by the finite, yet conservative sets (12) is called a B-spline relaxation. The major advantage is that B-spline relaxations avoid time gridding of the constraints, while they guarantee constraint satisfaction at all times. Since using relaxations comes down to replacing constraints on the spline by constraints on the control polygon, some conservatism is introduced, as shown on Figure 2 by the distance between the control polygon and the spline itself. This conservatism can be reduced by choosing a higher dimensional basis, at the cost of introducing extra constraints [19]. For a more detailed explanation, see [21].

Using a spline parameterization for the OCP, in combination with B-spline relaxations of semi-infinite constraints, leads to a tractable nonlinear optimization problem, which can be solved efficiently.

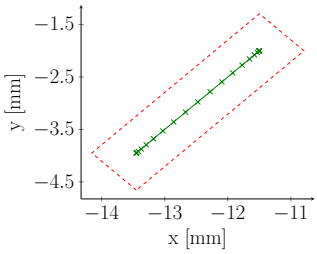
B. CNC trajectory generation

The presented spline-based trajectory generation method is implemented in OMG-tools: a user-friendly open-source toolbox. The Github page of this toolbox contains animations of simulations, and videos of the experimental validation [18]. OMG-tools is written in Python and uses CasADi [22] as a symbolic framework to formulate the trajectory generation problem, transform it into a tractable optimization problem and pass it to the solver. The default solver for all problems in this paper is Ipopt [23].

Regarding the specific implementation of the method in OMG-tools two practicalities demand special attention. The first practicality is that Ipopt is an interior point solver, and therefore requires a feasible initial guess to guarantee convergence to an optimal solution. In order to be feasible, the guess, which consists of values for the coefficients of the position splines, has to fulfill all constraints of the OCP. This means that the initial guess of the position trajectories has to lie inside the tolerance band. In addition, the initial guess also has to satisfy the kinematic limits of the tool axes, which is ensured by scaling its time axis accordingly. In a default step

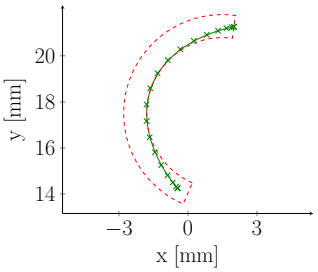


(a) Bang bang jerk trajectory guess

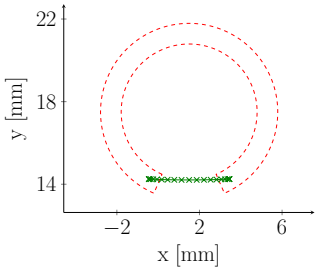


(b) Corresponding position trajectory guess

Fig. 3: Initial guess generation for a straight segment



(a) Optimized initial guess (feasible)



(b) Short-cut trajectory for wide ring segment

Fig. 4: Initial guess generation for a circle segment

of the receding horizon approach, a feasible guess for the first $N-1$ segments is readily obtained from the solution of the previous step. However, for the added segment, an initial guess needs to be computed separately.

For straight segments (G01), a bang-bang jerk profile is proposed to determine the initial guess of the position spline, by using a triple integration (see Figure 3a). As expected, this approach gives coefficients that lie on a straight line through the center of the tolerance band, as shown in Figure 3b.

Finding an initial guess for a circle segment (G02 and G03) is more involved, since the spline coefficients leading to a trajectory that lies inside the ring are not easily determined. Placing all coefficients on the center line of the ring does not necessarily lead to a trajectory that lies inside the tolerance band. Instead, the guess is obtained by solving a fitting problem that minimizes the jerk of the trajectory, while staying inside the tolerance band, as shown in Figure 4a. It is important to mention that B-splines cannot exactly represent a circle. As a consequence, for a given spline basis with a certain freedom (determined by the amount of knots and the degree), there is a lower bound on the tolerance for which a spline inside the ring segment exists. In practice, when using a degree three spline with 10 knot intervals, even for a tolerance of $2.5\mu m$ a feasible initial guess was found.

The second practicality is due to the fact that (6) actually constrains the trajectory to lie inside the complete ring, instead of inside the segment itself. As a consequence, for ring segments with a wide opening angle, the most optimal trajectory may actually pass along the other side of the ring segment (see Figure 4b). To avoid these trajectories, such wide ring segments are split in multiple segments. In practice, segments

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

TABLE I: Kinematic limits for the considered benchmarks

parameter	v_{max} $[\frac{m}{s}]$	a_{max} $[\frac{m}{s^2}]$	j_{max} $[\frac{m}{s^3}]$	ϵ [µm]
Figure 5	0.15	20	1500	1 or 500
Figure 8a	0.5	20	1420	2.5
Figure 8b	0.3	20	850	6
Figure 9	/	4	/	15

with an opening angle larger than 135° are split into two new segments with equal opening angles, as shown at the top of Figure 5.

In addition, because the continuity constraints between subsequent segments enforce the endpoint of segment j to lie in the overlap region between segments j and $j+1$, all trajectories stay inside their corresponding segment.

IV. Numerical validation

To evaluate the performance of the presented method, trajectories are computed for several benchmarks. For an anchor-shaped workpiece, the influence of the amount of simultaneously optimized segments on the obtained machining time is studied in detail. For other benchmarks, the resulting machining time is compared to the ones obtained by the method of [15] or to those computed with an industrial trajectory generator of Siemens [17]. Table I summarizes the applied kinematic limits and the allowed tolerance for the considered benchmarks. Since this is common for CNC machines, symmetric upper and lower kinematic limits were chosen. For all comparisons, spline trajectories of degree three with 20 knot intervals were selected. In addition, trajectories for three segments were optimized simultaneously in the receding horizon approach ($N = 3$), unless mentioned differently. All computations are made on a notebook with Intel Core i5-4300M CPU @2.60GHz x 4 processor and 8GB of memory.

A. Evaluation for an anchor-shaped workpiece

Figure 5 shows the machine tool trajectories that are computed for the time-optimal machining of the first benchmark: an anchor-shaped workpiece. It is clear how the trajectories efficiently exploit the tolerance band of $0.5\,\mathrm{mm}$ by cutting corners, hereby reducing the production time. For all simulations concerning the anchor, an extra feed rate constraint (4) was added, using an f_{max} of $150\,\frac{mm}{s}$.

To study the relationship between the tolerance (ϵ), the amount of simultaneously optimized trajectories (N), the obtained motion time (T_{mot}) and the required average and maximum calculation times for each step in the receding horizon approach (T_{avg} and T_{max}), Table II gives an overview of simulation results for different combinations of ϵ and N. This table shows that optimizing trajectories for only one segment at a time, thus requiring standstill at the end of each segment, gives a significantly higher motion time. This is because subsequent trajectories are decoupled in this case. Even when trajectories for only two subsequent segments are optimized simultaneously ($N = 2$), this decoupling effect almost completely vanishes: the obtained T_{mot} when putting $N = 2$ or when optimizing the trajectories for all segments

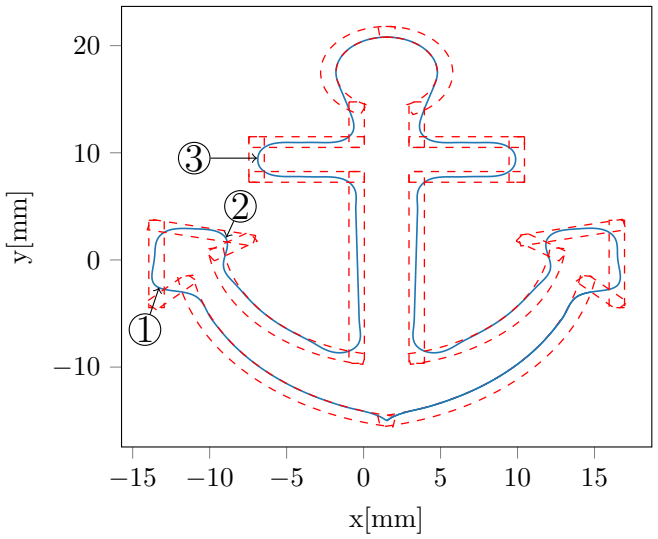


Fig. 5: Computed trajectories for an anchor-shaped workpiece with $\epsilon = 0.5\,\mathrm{mm}$

TABLE II: Effect of ϵ and N on the motion time (T_{mot}) and the maximum (T_{max}) and average (T_{avg}) computation time

$\epsilon = 1\,\mu m$	N	T_{mot} [s]	T_{avg} [ms]	T_{max} [ms]
	1	1.791	9	12
	2	1.703	34	54
	3	1.702	64	96
	24	1.702	641	641

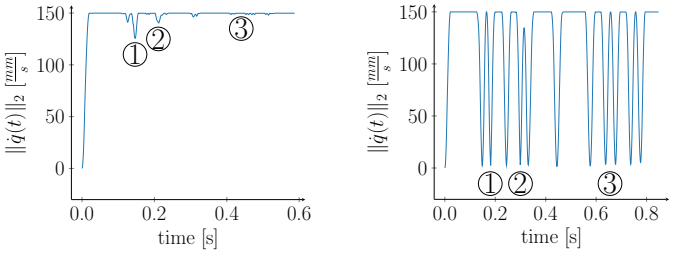
$\epsilon = 0.5\,mm$	N	T_{mot} [s]	T_{avg} [ms]	T_{max} [ms]
	1	1.754	10	19
	2	1.245	29	61
	3	1.176	43	64
	24	1.176	424	424

in one step ($N = 24$) hardly differs. On the other hand, the required solving times increase significantly for higher N values. This shows that optimizing trajectories for two or three segments simultaneously makes a good trade-off between optimality and computational cost. In addition, Table II shows that the effect of decoupling is smaller when the tolerance band is tighter. This is because fully decoupled trajectories form the optimal solution when ϵ is zero.

In addition, for the case in which $\epsilon = 1\,\mu m$ and $N = 3$, the required solving time for each step in the receding horizon is comparable with the machining time for the first segment in this horizon. This means that, with the appropriate preprocessing, the presented method is suited to use online, during machining.

To study the effect of the tolerance on the obtainable feed rate, the feed rate profiles for both tolerance values are compared. Figure 6 clearly shows that a larger tolerance allows for a more constant feed rate, while a lower tolerance often requires slowing down to stay inside the tolerance band. The numbers that are added to Figure 6 are linked to the equivalent numbers in Figure 5, indicating which feed rate is obtained at which point on the contour. In addition, Figure 7 shows the jerk trajectories for the x- and y-axis, which correspond to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics



(a) $\epsilon = 0.5\,\text{mm}$ (b) $\epsilon = 1\,\mu\text{m}$

Fig. 6: Feed rate profiles for half of the anchor

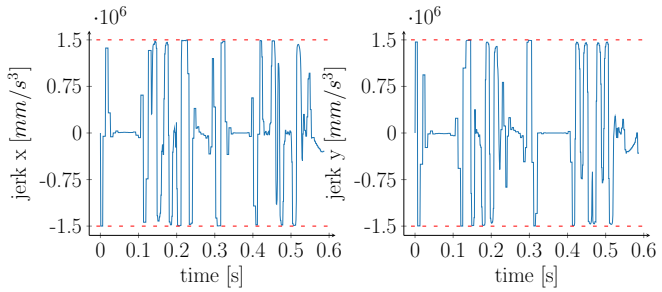


Fig. 7: Computed jerk trajectories for half of the anchor-shaped workpiece with $\epsilon = 0.5\,\text{mm}$

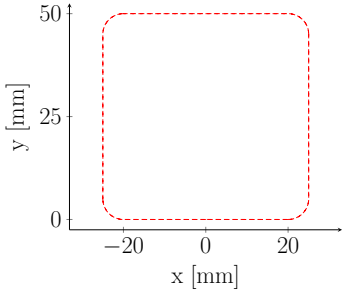
position trajectories of Figure 5.

Finally, additional constraints were included in the problem formulation to force the trajectory to pass exactly through all corner points of the contour, as is required by the method of [16]. Adding these constraints increases the required motion time from $1.176\,\text{s}$ to $1.385\,\text{s}$, which provides a strong indication that the requirement to pass exactly through the corner points leads to conservative trajectories.

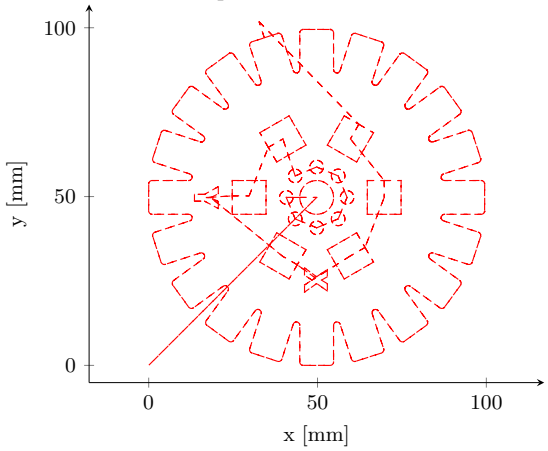
B. Comparison to alternative approaches

In order to quantify the performance of the spline-based trajectory generation approach, the current section compares it to several alternative methods. The first alternative method keeps the tolerance band, but demands exact standstill at the end of each GCode segment ($N = 1$). The second method uses an industrial trajectory generator from Siemens. The third method is presented in [15] and solves an OCP to compute trajectories that smoothly traverse corners in minimal time, while taking into account the kinematic constraints and path tolerance.

In the presented comparison, three benchmarks are considered. The first benchmark is a square with sides of $50\,\text{mm}$, of which the corners are rounded using circles with a radius of $5\,\text{mm}$, as shown in Figure 8a. The second benchmark is a complicated workpiece that contains a wide variety of shapes (see Figure 8b). This piece is designed to be machined by a laser cutter, explaining the straight connections between e.g. the different circles, on which the laser beam is turned off. The third benchmark is taken from [15] and consists of three corners with different opening angles, as shown in Figure 9. For all benchmarks, the corresponding limits are given in Table I.



(a) Square benchmark



(b) Complicated benchmark

Fig. 8: Studied benchmarks

TABLE III: Comparison of the motion times of different trajectory generation methods for the benchmarks of Figure 8

Benchmark:	Figure 8a ($\epsilon = 2.5\,\mu\text{m}$)	Figure 8b ($\epsilon = 6\,\mu\text{m}$)
Standstill-standstill	0.761 s	14.38 s
Siemens	0.64s	15.1 s
Spline-based	0.57 s	9.29 s

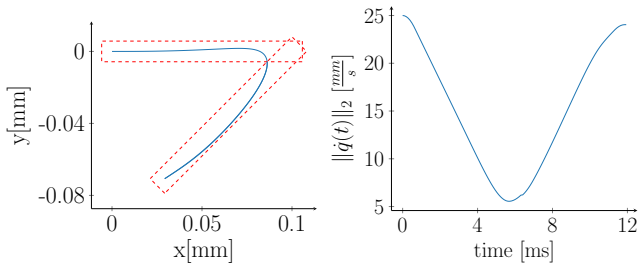
For the square-shaped benchmark, the second column of Table III summarizes the obtained total motion time of the tool for each method, showing that the presented method gives an 11% reduction in machining time compared to the industrial trajectory generator. The average computation time of a step in the receding horizon approach of the spline-based method is $58\,\text{ms}$, the maximum computation time is $78\,\text{ms}$.

For the complicated benchmark, the third column of Table III summarizes the obtained total motion time of the tool for each method, showing that the presented method gives a 38% reduction in machining time compared to the industrial trajectory generator. The average computation time of a step in the receding horizon approach is $63\,\text{ms}$, the maximum computation time is $156\,\text{ms}$. Note that the difference in motion time between both methods is more pronounced for the complicated workpiece, since it contains more segments, and therefore more opportunities to gain time. The GCode that describes the anchor and the two previous benchmarks can be found in [18].

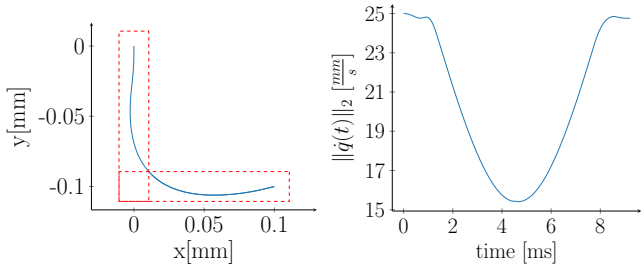
Finally, the presented spline-based trajectory generation method is compared to the approach presented in [15]. The au-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

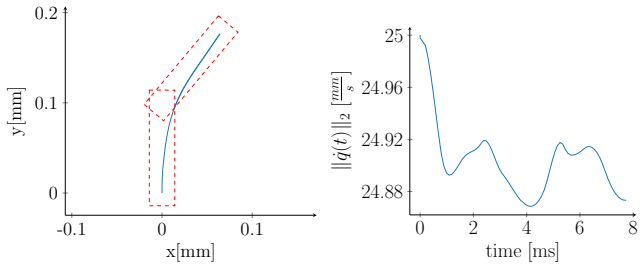
thors compute trajectories that smoothly traverse three corners with opening angles of respectively $45°$, $90°$ and $140°$. Table I gives an overview of the kinematic limits that are selected for this comparison. In addition, the feed rate is constrained to $25\,\frac{\text{mm}}{\text{s}}$. Because the authors of [15] define the tolerance as the maximum distance that is allowed between the corner point and the trajectory, the value that is selected for ϵ in the spline approach is tightened (depending on the specific angle) to fulfill this constraint. Figure 9 shows the computed position and corresponding velocity trajectories for the three considered corners.



(a) 45° angle



(b) 90° angle



(c) 140° angle

Fig. 9: Position and velocity spline trajectories obtained for different corners

Table IV makes a comparison of the motion times that are obtained with both methods for the three corners. As a reference, the motion time for exact corner following, requiring standstill in the corner point, is included in the first row of the table. Except for the $45°$ corner, for which the computed spline trajectory is 0.2% slower, trajectories that are 4% and 1% faster are obtained with the spline-based method. In addition, Table V shows that the presented method requires a solving time that is up to 45 times lower than the one required by the method from [15].

V. Experimental validation

To demonstrate the practical value of the presented trajectory generation approach, some of the discussed work-

TABLE IV: Comparison of the motion times for different corner smoothing trajectories obtained by the method presented in [15] and the proposed spline-based approach

Corner:	45°	90°	140°
Standstill-standstill	13.74 ms	14.66 ms	13.69 ms
Method [15]	11.89 ms	9.56 ms	7.81 ms
Spline-based	11.92 ms	9.21 ms	7.73 ms

TABLE V: Comparison of the solving times required to compute different corner smoothing trajectories with the method presented in [15] and the proposed spline-based approach

Corner:	45°	90°	140°
Method [15]	1.4 s	0.5 s	0.87 s
Spline-based	31.2 ms	32.8 ms	20.7 ms

pieces are produced on a 3-axis micro milling machine. First, Section V-A describes the machine in detail. Afterwards, Section V-B discusses the obtained results.

A. 3-axis micro-milling machine

Today's machine tools for manufacturing small workpieces, e.g. for the watch industry, are often weighting several tons and consuming more than $20\,\text{kW}$. A research program of the Applied University of Western Switzerland, under the lead of HE-Arc, developed a new concept for a micro machine, mainly intended for High Speed Machining (HSM) and Electrical Discharge Machining (EDM) [24]. The aspect ratio between machine and workpiece is small (5:1), and the power consumption is less than $2\,\text{kW}$, while keeping the same manufacturing performance as conventional production machines for small workpieces. Figure 10 shows the resulting 3-axis micro-milling machine, that is used for the experimental validation of the proposed trajectory generation method.

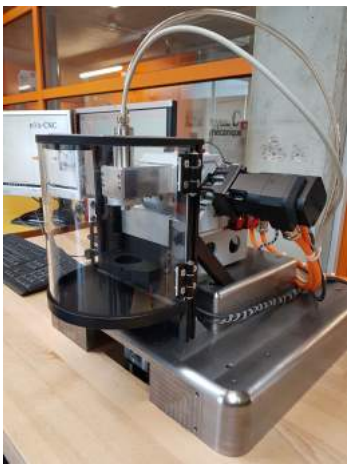


Fig. 10: 3–axis micro–milling machine

The machine design uses a stacked serial Cartesian configuration of the axes, driven by ball screws, creating a working space of $50 \times 50 \times 30$ mm. A high emphasis was put on a very light and highly rigid design: the sum of moving masses is only $10\,\text{kg}$, the static stiffness of the spindle holder is $\approx 5 \cdot 10^7\ \frac{\text{N}}{\text{m}}$ and the first eigenfrequency is about $180\,\text{Hz}$. The maximum

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2874617, IEEE Transactions on Industrial Electronics

velocity of the axes is $30\,\frac{\mathrm{m}}{\mathrm{s}}$, and the maximum acceleration is $20\,\frac{\mathrm{m}}{\mathrm{s}^2}$. The machine is equipped with a high speed spindle of $240\,\mathrm{W}$ that rotates at a maximum of 80.000 rpm. For the experimental validation explained below, a milling tool with a diameter of $1\,\mathrm{mm}$ was used to cut in a brass workpiece.

To steer the axes, high-end drives from Triamec are used, allowing a sampling frequency of $100\,\mathrm{kHz}$. The precalculated reference trajectories for position, velocity and acceleration are stored in a binary file. Afterwards, they are cut in chunks and transmitted in real time to the drives, using a PCI card with a large FIFO buffer, over a proprietary Trialink bus running at 10.000 samples per second. The desired setpoints are reached using standard cascaded PID control loops with feedforward compensations. At the same time, measurements from the incremental encoders are stored, enabling to quantify tracking errors for each axis.

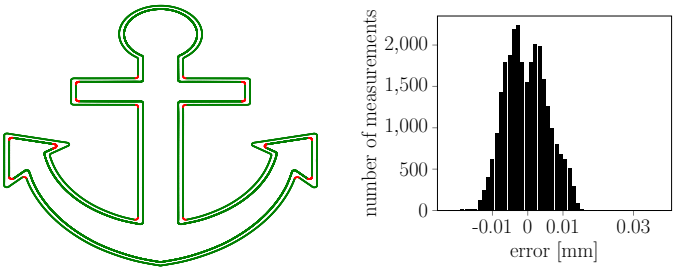
B. Experiments

The first machined workpiece is the anchor from Figure 5, selecting a maximum feed rate of $16\,\frac{\mathrm{mm}}{\mathrm{s}}$, a maximum acceleration of $20\,\frac{\mathrm{m}}{\mathrm{s}^2}$, and a maximum jerk of $1500\,\frac{\mathrm{m}}{\mathrm{s}^3}$. In order to study the effect of the selected tolerance value on the obtained trajectory and corresponding machining time, the workpiece is machined three times, reducing the tolerance from $0.5\,\mathrm{mm}$ over $0.1\,\mathrm{mm}$ to $0.01\,\mathrm{mm}$. Figure 11 clearly shows how the reduced freedom in the trajectory leads to a higher machining time.

(a) $\epsilon = 0.5\,\mathrm{mm}$, $T = 10.55\,\mathrm{s}$ (b) $\epsilon = 0.1\,\mathrm{mm}$, $T = 12.15\,\mathrm{s}$ (c) $\epsilon = 0.01\,\mathrm{mm}$, $T = 13.55\,\mathrm{s}$

Fig. 11: Comparison of machined anchors for increasing accuracy

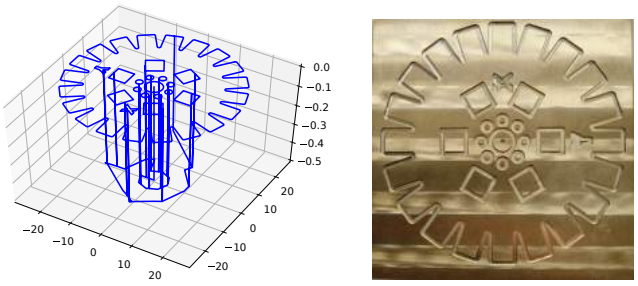
To analyze if the required tolerance for the workpiece of Figure 11c is reached, the obtained contour is measured with a tactile measurement machine. Because the machining tool has a larger radius than the measurement tool, undercut occurs in sharp corners. Therefore, it is not possible to measure the correct distance to the nominal contour at these points, and the measurements in this area are removed from the analysis (see Figure 12a). However, there are still plenty of measurements in the other corner points left in the data set. Comparing the measured positions with the desired contour shows an average error of $0.005\,\mathrm{mm}$ and a maximum error of $0.038\,\mathrm{mm}$. Note that while the average error is well below the tolerance, there are several measurements that violate the tolerance constraint of $0.01\,\mathrm{mm}$, as shown in Figure 12b. These violations have two main causes: (i) the computed trajectory exploits the tolerance band, such that its distance to the contour is equal to the tolerance at some points, leaving no margin for any deviation, and (ii) there are flexibilities between the motor encoders and the tool center point. In order to reduce these errors, the user can either reduce the allowed tolerance, or include the machine dynamics during the trajectory generation. The latter solution can be implemented in the proposed method by taking into account the determined resonance frequency of the flexibilities in the OCP.



(a) Indication of the removed measurements (red) in regions of undercut (b) Histogram of the measured contouring errors

Fig. 12: Analysis of the machined anchor workpiece using tactile measurements

The second machined workpiece is the one from Figure 8b. Originally, this workpiece is designed to machine with a laser cutter. During cutting, the laser beam is turned off on the straight segments that e.g. connect the different circles and squares. To simulate this behavior on a milling machine, the z-axis was retracted on connection segments (see Figure 13a). The same settings as for the anchors from Figure 11 were selected, using a tolerance of $0.01\,\mathrm{mm}$. With these settings, the required machining time becomes $52.1\,\mathrm{s}$. The trajectory



(a) (b)

Fig. 13: Retracting the z-axis on connection segments (a) to machine the complicated workpiece (b)

generation for retraction of the z-axis is implemented in OMG-tools as well. In addition, this toolbox also allows generating trajectories for workpieces that require multiple passes in the XY-plane, each time followed by a small movement in the z-direction. Movies of the machining of both workpieces can be found on the website of [18].

VI. Conclusion

Modern industry continuously strives to maximize the productivity of the available CNC machines. This paper presents a spline-based trajectory generation method that immediately takes into account the machine limits, process limits and

required accuracy. Therefore, it can exploit the allowed tolerance of the workpiece to compute trajectories that move the machine tool swiftly through corners, reducing the required production time to a great extent. Numerical simulations that compare the presented method to an industrial trajectory generator show an 11% reduction in machining time for a simple, and a 38% reduction for a more complicated workpiece. The comparison to a state-of-the-art corner smoothing approach shows that the presented method computes trajectories that are 0.2% slower to ones that are 4% faster, at computation times that are up to 45 times lower. An experimental validation on a 3-axis micro-milling machine proves the practical applicability of the presented method. Generating spline-based trajectories for a specific workpiece is facilitated by OMG-tools, a user-friendly open-source software toolbox.

REFERENCES

[1] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - An open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[2] A. V. Rao, D. A. Benson, C. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington, "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, vol. 37, no. 2, pp. 1–39, 2010.

[3] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," *Fast Motions in Biomechanics and Robotics*, 2005.

[4] F. Debrouwere, "Optimal Robot Path Following: Fast Solution Methods for Practical Non-convex Applications," Ph.D. dissertation, KU Leuven, 2015.

[5] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[6] M. Steinlin, "Model based Feed-rate Optimization for Machine Tool Trajectories," Ph.D. dissertation, ETH Zürich, 2013.

[7] Q. Zhang, S. Li, and J. Guo, "Smooth time-optimal tool trajectory generation for CNC manufacturing systems," *Journal of Manufacturing Systems*, vol. 31, no. 3, pp. 280–287, 2012.

[8] F. Sellmann, "Exploitation of tolerances and quasi-redundancy for set point generation," Ph.D. dissertation, ETH Zürich, 2014.

[9] T. C. Lu, S. L. Chen, and E. C. Y. Yang, "Near Time-optimal S-curve Velocity Planning for Multiple Line Segments under Axis Constraints," *IEEE Transactions on Industrial Electronics*, 2018.

[10] J. W. Jeon and Y. Y. Ha, "A Generalized Approach for the Acceleration and Deceleration of Industrial Robots and CNC Machine Tools," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 1, 2000.

[11] K. M. Nittler and R. T. Farouki, "Efficient high-speed cornering motions based on continuously-variable feedrates. II. Implementation and performance analysis," *International Journal of Advanced Manufacturing Technology*, vol. 88, no. 1-4, pp. 159–174, 2017.

[12] Y. Altintas and K. Erkorkmaz, "Feedrate Optimization for Spline Interpolation In High Speed Machine Tools," *CIRP Annals - Manufacturing Technology*, vol. 52, no. 1, pp. 297–302, 2003.

[13] N. Van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation," *Proceedings of the 2016 European Control Conference*, no. September, pp. 477–482, 2016.

[14] Y.-C. Chang, C.-W. Chen, and T.-C. Tsao, "Near Time-Optimal Real-Time Path Following Under Error Tolerance and System Constraints," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 7, 2018.

[15] M. Duan and C. Okwudire, "Minimum-time cornering for CNC machines using an optimal control method with NURBS parameterization," *International Journal of Advanced Manufacturing Technology*, vol. 85, no. 5-8, pp. 1405–1418, 2016.

[16] P. Bosetti and E. Bertolazzi, "Feed-rate and trajectory optimization for CNC machine tools," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 6, pp. 667–677, 2014.

[17] Siemens, "Sinumerik," Tech. Rep., 2005.

[18] R. Van Parys and T. Mercy, "OMG-tools," https://github.com/meco-group/omg-tools, 2016.

[19] W. Van Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mechanical Sciences*, vol. 6, no. 2, pp. 163–171, 2015.

[20] C. de Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[21] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *European Control Conference (ECC)*, pp. 1586–1591, 2016.

[22] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi - A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[23] L. T. Biegler and A. Wächter, "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[24] Haute Ecole Arc, "Machine à outils à cinq axes," 2017, wO2017182963.

Tim Mercy received the M.Sc. degree in mechanical engineering, with a specialization in mechatronics and robotics, from the KU Leuven (Belgium) in 2014. Currently, he is working at KU Leuven towards the Ph.D. degree.

His research interests include optimal motion planning for autonomous systems, with a focus on vehicles moving through industrial warehouses and systems operating in agricultural environments. In addition, he is interested in optimal control of CNC machines.

Nicolas Jacquod received his B.Sc. degree in microtechnology in 2016, from the University of Applied Sciences Western Switzerland, Yverdon-les-Bains, Switzerland. He is currently working as R&D assistant at the institute of mechanics of HEIG-VD.

His research interests contain the analysis of vibrations in various types of machines and numerical control of machine tools.

Raoul Herzog studied at ETH Zürich where he obtained the Diploma of Electrical Engineering and the Ph.D. degree in 1986 and 1991 respectively. From 1992 to 2003 he worked with the Swiss magnetic bearing manufacturer MECOS Traxler AG in Winterthur. In 2004, he joined the University of Applied Sciences in Yverdon-les-Bains. He is affiliated to the Institute of Industrial Automation and is steering committee member of the Swiss Society for Automatic Control SGA-ASSPA.

His research interests are in the areas of mechatronics, in particular modeling, trajectory optimization and control.

Goele Pipeleers is an assistant professor at the Department of Mechanical Engineering of the KU Leuven. She received her M.Sc. degree in mechanical engineering and her Ph.D. degree in mechanical engineering from the KU Leuven, in 2004 and 2009, respectively. She has been a Post-doctoral Fellow of the Research Foundation Flanders, and a visiting scholar at the Colorado School of Mines and at the University of California Los Angeles.

Her research interests include convex optimization, optimal and robust control, and their applications in mechatronics.