Time optimal trajectory generation from polyline with velocity, acceleration and spatial deviation constraints

A Şerban

1Mechanical Engineering, “Gheorghe Asachi” Technical University of Iasi, Iasi, Romania

E-mail: ser.alexandru@gmail.com

Abstract. In this paper is presented a method for generating continuous and differentiable trajectory from an input path described by a polyline while maintaining accelerations and velocities constraints. The method is based on replacing the path corners with parabolic blends of constant acceleration. The complete spatial control of the resulted trajectory is obtained with a velocity reduction procedure which ensures that the trajectory doesn’t deviate from the input path further than a settable value. The total time is minimized by replacing the non-maximized constant velocity sections with boost sequences containing maximized symmetrical accelerating and decelerating time frames while also keeping the velocity constraints. The method was successfully tested in practice and the results were very good even on complex input paths.

1. Introduction

The manufacturing capacity and capability have dramatically increased starting from the industrial revolution. For sure a very important place in this sharp development belongs to industrial robots and CNCs. A lot of interest was shown towards the execution time optimization of the jobs transferred from human to machine because even a very small percentage improvement can have a major impact to the cost. But, compared to human flexibility even in realizing very simple jobs a robot meets a lot of problems like obstacle collision which should be avoided, time optimal trajectory selection, mechanical constraints, etc.

There are a lot of studies dedicated to path planning [1-7] and trajectory generation [8-15]. Sometimes the output of path planning algorithms consists of a polyline defined by a set of waypoints connected by line segments. Obviously, the output polyline isn’t differentiable at the waypoints. This fact leads to the requirement that the moving robot or CNC axis to stop at the waypoints in order to change the direction due to the physical/mechanical limitations (velocity, acceleration and jerk constraints). Even if in this way the path is traversed exactly this approach is unacceptable suboptimal in term of execution time. A typical example which illustrates this inconvenient is represented by polylinearized curves which contain a lot of small line segments.

In this paper we propose a simple method and relatively easy to implement which transforms a polyline to a differentiable trajectory by replacing the initial path with parabolic curves at the waypoints while maintaining the velocity and acceleration constraints. This procedure leads to spatial deviations from the original input path which are controlled and limited to a settable values. The total time is minimized by choosing an optimal timing for the parabolic replacements and by replacing the non-maximized velocity sections with boost sequences. Our method was successfully tested in...
practice for a lot of paths and the results are stable even when the input path contains extreme turning waypoints.

2. The algorithm

Our algorithm to convert a given path described by a polyline to an optimal differentiable trajectory is based on the method from [16] which involves using parabolic blends [17-18]. Our main contribution consists of inserting the so called boost sequences and additional constraints like spatial deviation.

The algorithm can be divided in the following phases which will be described in the next sections:

1. Generate the raw trajectory - using parabolic blends;
2. Remove the time frames overlaps – if any;
3. Ensure the spatial deviation limitation;
4. Replace non-maximized constant velocity sections by boost sequences.

2.1. Generate the raw trajectory

Consider the figure 1 [16] which illustrates a one-dimensional path.

![Diagram of a one-dimensional path](image)

**Figure 1.** The raw trajectory for one-dimensional path [16].

The quantities from figure 1 represent:
- \( n \) – the number of waypoints in the input path;
- \( q_i \) with \( i = 1, n \) – the waypoints which describe the input path;
- \( \Delta T_i \) with \( i = 1, n - 1 \) – the duration to move from \( q_i \) to \( q_{i+1} \);
- \( t^b_i \) with \( i = 1, n \) – the duration of the blend phase associated with \( q_i \) – constant non-zero acceleration phase centered around the waypoints with respect to time;
- \( t^f_i \) with \( i = 1, n - 1 \) – the duration of the constant velocity phase between \( q_i \) and \( q_{i+1} \) – zero acceleration phase.

\[
t^f_i = \Delta T_i - \frac{1}{2} (t^b_i + t^b_{i+1}) \tag{1}
\]
- \( v_i \) with \( i = 0, n \) – the constant velocity associated with \( t_i^i \) phase. \( v_0 \) and \( v_n \) are artificially added as initial and final velocities (regularly equal to 0).

It can be observed that the raw trajectory consists of an alternation of blend phases and constant velocity phases. Assuming that \( \Delta \tau_l \) and \( t_i^b \) are known it can be computed the velocities and accelerations using 2 and 3.

\[
v_i = \frac{q_{i+1} - q_i}{\Delta \tau_l} \tag{2}
\]

\[
a_i = \frac{v_i - v_{i-1}}{t_i^b} \tag{3}
\]

The total duration \( t_f \) is given by 4.

\[
t_f = \frac{t_i^b}{2} + \sum_{i=1}^{n-1} \Delta \tau_l + \frac{t_i^b}{2} \tag{4}
\]

The raw trajectory and its derivatives can be defined using 5, 6 and 7, respectively.

\[
q(t) = \begin{cases} 
q_i + v_{i-1} (t - T_i) + \frac{1}{2} a_i \left( t - T_i + \frac{t_i^b}{2} \right)^2, & T_i - \frac{t_i^b}{2} \leq t \leq T_i + \frac{t_i^b}{2} \\
q_i + v_i (t - T_i), & T_i + \frac{t_i^b}{2} \leq t \leq T_{i+1} - \frac{t_{i+1}^b}{2}
\end{cases} \tag{5}
\]

\[
\dot{q}(t) = \begin{cases} 
q_{i-1} + a_i \left( t - T_i + \frac{t_i^b}{2} \right), & T_i - \frac{t_i^b}{2} \leq t \leq T_i + \frac{t_i^b}{2} \\
v_i, & T_i + \frac{t_i^b}{2} \leq t \leq T_{i+1} - \frac{t_{i+1}^b}{2}
\end{cases} \tag{6}
\]

\[
\ddot{q}(t) = \begin{cases} 
a_i, & T_i - \frac{t_i^b}{2} \leq t \leq T_i + \frac{t_i^b}{2} \\
0, & T_i + \frac{t_i^b}{2} \leq t \leq T_{i+1} - \frac{t_{i+1}^b}{2}
\end{cases} \tag{7}
\]

At this moment the unknowns are \( \Delta \tau_l \) and \( t_i^b \) which are computed by maximizing the velocities \( v_i \) and blend accelerations \( a_i \) according to the constraints. The initial constraints for the velocity components are denoted as \((v^j max > 0)\), the resultant velocity constraint is denoted as \((v^r max > 0)\), the constraints for the acceleration components are denoted as \((a^j max > 0)\) and the resultant acceleration constraint is denoted as \((a^r max > 0)\), where \( j \) is the component index.

\( \Delta \tau_l \) is computed as:

\[
\Delta \tau_l = \max(\Delta \tau_{l^c}, \Delta \tau_{l^t}) \tag{8}
\]

where:

\[
\Delta \tau_{l^c} = \max_j \left( \frac{||q_{i+1} - q_i||}{v^j max} \right) \tag{9}
\]

\[
\Delta \tau_{l^t} = \frac{||q_{i+1} - q_i||}{v^r max} \tag{10}
\]

\( v^j max \) is the velocity constraint for the \( j \) component between \( q_i \) and \( q_{i+1} \). For the raw trajectory computation \( v^j max \) are all equal to the initial velocity constraints \((v^j max)\) and will be eventually reduced in the removing overlaps phase (2) and/or in the ensuring spatial deviation phase (3). The total velocity constraint \( v^r max \) is important for example in the cases when a blade cutting tool has a frequency limitation or a laser cutting tool has a power limitation. Both previous examples restrict the moving velocity \( v^r max \).
$t^b_i$ is computed as:

$$t^b_i = \max(t^{bc}_i, t^{pt}_i, t^{b\theta}_i)$$ \hspace{1cm} (11)

where:

$$t^{bc}_i = \max_j \left( \frac{|v^l_i - v^l_{i-1}|}{a/\text{max}} \right)$$ \hspace{1cm} (12)

$$t^{pt}_i = \frac{|v^l_i - v^l_{i-1}|}{a^b\text{max}}$$ \hspace{1cm} (13)

For the case when a rotating blade tool is considered we have an additional acceleration constraint $t^{b\theta}_i$.

$$t^{b\theta}_i = 2 \sqrt{\frac{\Delta \theta}{a^b\text{max}}}$$ \hspace{1cm} (14)

Where $\Delta \theta$ represent the blade rotation from $q_{i-1}q_i$ segment orientation to $q_iq_{i+1}$ segment orientation and $a^b\text{max}$ is the rotating tool acceleration constraint. Relation (14) is obtained considering zero velocity for the rotating tool before and after the parabolic blend phase and same magnitudes and opposite signs for the rotating tool accelerations for the first and second half of the parabolic blend phase.

The raw trajectory obtained with the above procedure can eventually have discontinuities or may not be differentiable or may not satisfy the spatial deviation constraint. These situations will be treated in the next two phases of the proposed algorithm.

### 2.2 Remove the time frames overlaps

The trajectory time frames should satisfy relation (15) which means that the blending phases should not overlap with constant velocity phases in respect to time.

$$t^b_i + t^b_{i+1} \leq 2\Delta T_i$$ \hspace{1cm} (15)

Relation (15) should be satisfied for the entire trajectory in order to avoid discontinuities and to ensure the differentiability.

We propose a different method than the original suboptimal approach in [16] to remove the time frames overlaps - if any – which results in a lower total duration $T_f$. Even the computational cost is bigger our method is easy to implement and the execution time is neglectable. The procedure is simple and is described by the following pseudocode.

```plaintext
% Pseudocode for the removing time frames overlaps procedure

While hasOverlaps(trajectory)
    idx = getFirstIndexOverlap(trajectory)
    applyVelocityReduction(idx, f)
    regenerateTrajectory(trajectory)
end
```

- Function hasOverlaps(trajectory) returns TRUE if relation (15) is not satisfied and FALSE otherwise;
- Function getFirstIndexOverlap(trajectory) returns the first index which violates relation (15);
- Function applyVelocityReduction(idx, f) reduce all the velocity components constraints by multiplying with a factor of $f$ for $v^l_{dx,\text{max}}$ and $v^l_{dx-1,\text{max}}$ if their resultant velocity have the same magnitude. If the magnitudes are different then the velocity reduction is applied only for
the velocity components of the resultant velocity with the biggest magnitude. In our experiments we use a value of 0.99 for \( f \).

- Function \( \text{regenerateTrajectory}(\text{trajectory}) \) regenerates the values of \( \Delta T, t^b, v, a \) affected by the velocity reduction (from \( idx - 1 \) to \( idx + 1 \)).

2.3. Ensure the spatial deviation limitation
We define the spatial deviation as the distance from a waypoint to its associated parabolic blend. It is possible that the spatial deviation of certain waypoints to be unacceptable big. For example a big deviation may cause the collision between a moving robot and an obstacle. The reference paper [16] just mentions this fact without providing a spatial deviation control method. In this respect we developed a procedure very similar with the removing time frames overlaps procedure which is described by the next pseudocode.

\% Pseudocode for the procedure to limit the spatial deviation
While hasUnacceptableDeviation(\text{trajectory, dev})
  \quad idx = getFirstIndexUnaccetableDeviation(\text{trajectory, dev})
  \quad applyVelocityReduction(idx, f)
  \quad regenerateTrajectory(\text{trajectory})
end

- Function \( \text{hasUnaccepatbleDeviation}(\text{trajectory, dev}) \) returns TRUE if the trajectory has any unacceptable spatial deviation and FALSE otherwise. The parameter \( \text{dev} \) represents the maximum accepted deviation.
- Function \( \text{getFirstIndexUnaccetableDeviation}(\text{trajectory, dev}) \) returns the first index where the spatial deviation violates the constraint to be smaller than \( \text{dev} \);
- Functions \( \text{applyVelocityReduction}(idx, f) \) and \( \text{regenerateTrajectory}(\text{trajectory}) \) are the same described at the previous phase of the algorithm.

Computing the spatial deviation at a certain waypoint is relatively easy. The parabolic blend trajectory can be extracted from (5).

\[
q^b_i(t) = q_i + v_{i-1}(t - T_i) + \frac{1}{2}a_i\left(t - T_i + \frac{t^b_i}{2}\right)^2, T_i - \frac{t^b_i}{2} \leq t \leq T_i + \frac{t^b_i}{2}
\]  
\tag{16}

The squared distance function from the waypoint \( q_i \) to \( q^b_i(t) \) is defined as:

\[
D_s(t) = \|q^b_i(t) - q_i\|^2 = \left|v_{i-1}(t - T_i) + \frac{1}{2}a_i\left(t - T_i + \frac{t^b_i}{2}\right)^2\right|^2
\]  
\tag{17}

From (17) results that the squared distance function is a polynomial function in \( t \) of order 4. The spatial deviation can be obtained by finding the critical points of this function in the interval \( T_i - \frac{t^b_i}{2} \leq t \leq T_i + \frac{t^b_i}{2} \). Usually the minimization process provides one real solution and 2 complex solutions for \( t \). After the solution is found it can be computed the spatial deviation and compared with \( \text{dev} \).

2.4. Replace constant velocity sections by boost sequences
In the original paper [16] the constant velocity section is traversed with maximized velocity except the case when velocity reduction is applied due to time frames overlaps which results in a high suboptimal solution. Unfortunately this is the prevalent case in practice especially when polyliniarized curves are involved in the input path. In our approach velocity reduction is also used in ensuring spatial deviation
phase which increase the effect of this behaviour. However, due to the physical capability (accelerations limitation) this behaviour cannot be eliminated but it can be reduced by replacing the suboptimal constant velocity sections (with non-maximized velocity) by a boost sequence. The idea is to apply a maximized acceleration and deceleration such that the section is traversed in the shortest time while maintaining the velocity constraints and keeping the velocity at the end of the boost sequence equal to the initial velocity in order to not affect the next blend section.

In this respect the constant velocity section will be replaced by a boost sequence containing:
- Two time frames (acceleration and deceleration) – when the maximized velocity cannot be reached due to the short duration of the section;
- Three time frames (acceleration, 0 and deceleration) – when the maximized velocity can be reached and maintained during the 0 acceleration time frame.

The procedure for replacing a suboptimal constant velocity \(v_i\) section with an optimized boost sequence is described below:
- Compute the maximized acceleration \(a\) such that
\[
a = \lambda \cdot v_i
\] (18)
The reason that the acceleration should have this form is to keep the trajectory unchanged on the section.
- Compute the time \(t_{\text{mid}}\) to the middle of the section using the maximized acceleration.
- Compute the time \(t_{\text{max}}\) to achieve the maximized velocity using the maximized acceleration.
- If \(t_{\text{max}} \geq t_{\text{mid}}\) then replace the section with a boost sequence containing two time frames of duration \(t_{\text{mid}}\) and accelerations \(a\) and \(-a\) respectively.
- If \(t_{\text{max}} < t_{\text{mid}}\) then replace the section with a boost sequence containing three time frames of duration \(t_{\text{mid}}, t_{\text{const}}, t_{\text{mid}}\) and accelerations \(a, 0\) and \(-a\) respectively. Here \(t_{\text{const}}\) is the duration of the constant maximized velocity frame and can be easily computed.

3. Results
The method presented in this paper was successfully tested in practice on a couple of leather cutters for more than 1000 different parts which were cut. The results were very good and some examples are presented in the next figures.

The leather cutters have a rotating blade tool and \(x, y\)-axis displacements. The velocity constraints were set to \(v^x_{\text{max}} = v^y_{\text{max}} = v^t_{\text{max}} = 1 \text{ m/s}\) which means that \(v^t_{\text{max}}\) dominates the velocity maximization. This is normal due to the blade cutting frequency limitation. The acceleration constraints were set to \(a^x_{\text{max}} = a^y_{\text{max}} = 3 \text{ m/s}^2\) and \(a^t_{\text{max}} = \sqrt{18} \text{ m/s}^2\) which means that acceleration components dominates the acceleration maximization. This is also normal due to the axis electrical motors capabilities. The accepted spatial deviation were set to \(\text{dev} = 1 \text{ mm}\) and the reduction factor to \(f = 0.99\).

In figure 2 are shown the results for a square path with the edge length equal to 1 m: figure 2a shows the input path, the waypoint and the resultant trajectory, figure 2b shows zoomed caption at the corners and figure 2c shows the velocities profile. In figure 2b it can be observed that the spatial deviation at the start-stop waypoint is 0.
Figure 2. Results for a square path: (a) the input path and resultant trajectory (b) zoomed caption at corners (c) velocities profile.

In figure 3 are shown the results for a circle with 1 m diameter described by a polyline with 128 equidistant points: figure 3a shows the input path, the waypoint and the resultant trajectory and figure 3b shows the velocities profile.

In figure 4 are shown more examples of parts that were cut using our method.
Figure 3. Results for a circle: (a) the input path and resultant trajectory (b) velocities profile.

Figure 4. Some examples of input paths and resultant trajectories.
In figure 5 is shown an example of a path with an extreme turning: figure 5a shows the input path, the waypoints and the resultant trajectory and figure 5b shows a zoomed caption to the extreme turning waypoint.

4. Conclusion
In this paper we presented a method for generating continuous and differentiable trajectory from an input path described by a polyline while maintaining accelerations and velocities constraints. Also, the proposed method incorporates a procedure to ensure that the generated trajectory doesn’t deviate from the input path further than a settable value. This fact allows a complete spatial control of the trajectory. Additionally, the non-maximized constant velocity sections are replaced with boost sequences in order to minimize the total duration. The method was successfully tested in practice on a couple of leather cutters for a large set of parts which were effectively cut. The results were very good and the method showed great stability even on parts containing extreme turning waypoints.

5. References
[1] Wall D G, Economou J, Goyder H, Knowles K, Silson P and Lawrance M 2015 Mobile robot arm trajectory generation for operation in confined environments Journal of systems and control engineering 229 pp 215-234
[2] Katzschmann R, Kroger T, Asfour T and Khatib O 2013 Towards online trajectory generation considering robot dynamics and torques limitations Proceedings of the 2013 IEEE/RSJ International Conference on robots and systems pp 5644-5651
[3] Guarino Lo Bianco C and Ghilardelli F 2012 Third order system for the generation of minimum-time trajectories with asymmetric bounds on velocity, acceleration and jerk Proceedings of the 2012 IEEE International Conference on intelligent robots and systems pp 137-143
[4] Khansari-Zadeh S-M and Billard A 2012 A dynamical system approach to realtime obstacle avoidance Autonomous robots 32 pp 433-454
[5] Flacco F, Kroger T, De Luca A and Khatib O 2012 A depth space approach to human-robot collision avoidance Proceedings of the 2012 IEEE International Conference on robotics and automation ICRA 2012 pp 338-345
[6] Balan L and Bone G M 2006 Real-time 3D collision avoidance method for safe human and
robot coexistence Proceedings of the 2006 IEEE/RSJ International Conference on intelligent robots and systems IROS 2006 pp 276-282

[7] Ramos F, Gajamohan M, Huebel N and D’Andrea R 2013 Time-Optimal online trajectory generator for robotic manipulators Institute for Dynamics systems and control ETH Zurich

[8] Zhang K, Yuan C-M, Gao X-S, Li H-B 2012 A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk Robotics and Computer-Integrated Manufacturing 28 pp 472-483

[9] Zhang K, Guo J-X and Gao X-S 2013 Cubic spline trajectory generation with axis jerk and tracking error constraints Precision Engineering and Manufacturing 14 pp 1141-1146

[10] Lee A-C, Lin M-T, Pan Y-R, Lin W-Y 2011 The feedrate scheduling of NURBS interpolator for CNC machine tools Computer-Aided Design 43 pp 612-628

[11] Gasparetto A and Zanotto V 2008 A technique for time-jerk optimal planning of robot trajectories Robotics and Computer-Integrated manufacturing 24 pp 415-426

[12] Sencer B, Altintas Y and Croft E 2008 Feed optimization for five-axis CNC machine tools with drive constraints International journal of machine tools and manufacture 48 pp 733-745

[13] Lay J-Y, Lin K-Y, Tseng S-J, Ueng W-D 2008 On the development of a parametric interpolator with confined chord error, federate, acceleration and jerk International journal of advance manufacturing technology 37 pp 104-121

[14] Emami M-M and Arezoo B 2010 A look-ahead command generator with control over trajectory and chord error for NURBS curve with unknown arc length Computer-Aided Design 42 pp 625-632

[15] Beudaert X, Lavernhe S and Tournier C 2012 Feedrate interpolation with axis jerk constraints on 5-axis NURBS and G1 tool path International Journal of Machine Tools and Manufacture 57 pp 73-82

[16] Kunz T and Stilman M 2011 Turning paths into trajectories using parabolic blends Technical report at Georgia Institute of Technology

[17] Craig J J 2004 Introduction to Robotics: Mechanics and Control (Upper Saddle River: Pearson Prentice Hall, 3rd edition)

[18] Siciliano B, Sciavicco L and Villani L 2009 Robotics: modeling, planning and control (Springer)