A New Method for Generating Work Piece Surface Representations for Robotic Machining

Leth, Nikolaj Westphal; Petersen, Henrik Gordon

Published in:
2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

DOI:
10.1109/IROS51168.2021.9636774

Publication date:
2021

Document version:
Accepted manuscript

Citation for published version (APA):

Go to publication entry in University of Southern Denmark's Research Portal

Terms of use
This work is brought to you by the University of Southern Denmark. Unless otherwise specified it has been shared according to the terms for self-archiving. If no other license is stated, these terms apply:

• You may download this work for personal use only.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying this open access version.

If you believe that this document breaches copyright please contact us providing details and we will investigate your claim. Please direct all enquiries to puresupport@bib.sdu.dk
A New Method for Generating Work Piece Surface Representations for Robotic Machining

Nikolaj W. Leth and Henrik G. Petersen

Abstract—Execution of automatically generated programs for accurate robotic machining requires the generated trajectories to be not only accurate with respect to the work piece, but also that the trajectories are continuous differentiable ($C^1$) while avoiding unnecessary large curvatures leading to large accelerations that could compromise machining quality or speed. A widely used work piece representation is 3D triangle meshes as they can be easily generated in any CAD representations and from surface scans, and they are also very suitable for robotics applications. However, they lack the $C^1$ property across the triangle edges.

In this paper, a new method for generating $C^1$ surfaces based on 3D triangle meshes is presented. It will be shown by an example that the method is as good as existing methods with respect to the accuracy of the generated surface, and that the problem with large curvatures is much smaller than for existing methods. Moreover, the difficult input specification of derivatives at the vertices is avoided with this method.

I. INTRODUCTION

Robotic surface treatment has been a large topic within the community for many years: from non-contact processes such as spray painting, to in-contact processes such as polishing and different forms of machining. Especially in-contact processes are highly dependent on the smoothness of the generated trajectories. In this context, smooth trajectory generation for robot machining is moving away from being handled by the native robot controller to be handled in the offline programming environment so that the robot programmer has more control over the final trajectory. Performing this type of programming depends on the quality and smoothness of the CAD model for the trajectory generation to produce good results, and at least a $C^1$ (first-order parametric continuity) CAD surface is needed. In many robotic applications, the discrete triangle meshes described in STL files are the preferred model representations over more descriptive formats such as STEP, as it makes for simple scene creation, collision detection and visualization. Moreover, triangle meshes can be generated from any CAD formats or surface scans, and are hence, by far, the most widespread representation. However, it lacks the important property of $C^1$ which is required for programming in contact surface treatment processes.

In a collaborative project with the Danish company Bang & Olufsen FACTORY 5, the aim is to automate the complete procedure starting with a work piece represented as a triangle mesh of surfaces to the generation of machining trajectories and online correction of these. The first step is to represent the surface in a suitable way, which leads to the following desired properties: 1) The surface must be continuous differentiable ($C^1$) everywhere while avoiding artificial high curvatures. Not satisfying this property would lead to curves where machining speed would need to be reduced to sustain the accuracy, hence leading to a decreased throughput and sometimes worse quality. 2) Derivatives at the vertices are difficult to estimate correctly and might be error-prone in irregular meshes, and hence these should not be a requirement for the method. 3) Triangle meshes have different sizes and shapes when curvature sizes are different. The resulting surface must capture this. A simple way to ensure this is to require that all triangle vertices must reside on the surface and changes of the mesh are disallowed. 4) Over-parametrization of the surface (e.g. too dense grids) should be avoided as this would lead to unnecessary complex representations and increased difficulties in online adjustment of trajectories.

In Computer-Aided Design (CAD), Bézier, B-spline and NURBS (Non-Uniform Rational B-Spline) patches [1], are the de facto standard for generating mathematical descriptions of amorphous surfaces. A logical approach would therefore seemingly be to create a parametric (Bézier, B-Spline or NURBS) surface using a global interpolation. The global interpolation would ensure that we include the exact vertex positions from the triangle mesh, while allowing us to obtain at least $C^1$ as described in [1]. However, as...
meshes are often highly non-uniform, this approach is not realizable as global interpolation requires the given points to be interpolated by orthogonal grid lines. Other studies, e.g. [2], [3], have focused on local interpolation, in which a number of smaller local patches are created and stitched together. With such an approach all of the points do not need to be part of the same orthogonal grid. However, often the mesh have to be remeshed for the fitting to work, which will often spoil fitting to the exact vertex points from the original mesh.

To overcome the need for the points to be arranged in an orthogonal grid, approximation methods for fitting the surfaces are considered. In its simplest form, an approximation process will start by fitting a surface with a few control point using, for example, a least squares approach. If the error of the fitted surface is too high, a new surface with more control points / knots or of a different degree would then be tried, repeatedly, until the fitting error is below a set threshold [1]. This process can lead to a high number of control points and different artifacts when dealing with more complex surfaces [4]. More advanced approximation methods have therefore been proposed, e.g. by adding additional criteria [5], using discrete stationary wavelet transform [6], genetic algorithms [7], evolutionary strategies [8], [9], combinations of hybrid optimization algorithms and iterative schemes [10], [11], subdivision methods [12], reparametrization [13]. However, in general these methods also suffer from not fitting the mesh vertices and many of them also over parameterizes the surface.

A promising approach, which has been explored within finite element methods and the computer graphics communities, is Bézier triangles. As the name indicates, the Bézier triangles use a triangle form, compared to the formerly described square or rectangular patch forms. This directly allows the use of interpolation methods on the triangle mesh by choosing the Bézier triangles as the same as the mesh triangles. Hence, all vertices will automatically be on the surface. However, the aspect of $C^1$ continuity is still a concern, as the resulting constraints for Bézier triangles are significantly more difficult to handle.

In this paper, we will present a new method for creating a surface from a triangle mesh using cubic Bézier triangles requiring only vertex positions as input and no derivative estimates. A global convex polygon-shaped, piecewise linear interpolation will be created alongside, which allow us to satisfy the constraints for smoothness, $C^1$, across the boundaries between the Bézier triangles. We will show that our method is as precise as two of the most widely used state of the art approaches where vertex derivatives are required. Moreover, we will show that the resulting curvatures are much less critical making our method more suitable for robotic machining.

In Section II, we summarize the related work on Bézier triangles including the difficulties with the existing methods. In Section III, the basic structure of the Bézier triangles is reviewed together with a common method for creating a $C^0$ surface from vertex points and normal vectors. In Section IV, the constraints for continuity across boundaries between Bézier triangles are described along with a method for creating a global parametrization and how this parametrization can be used to move the internal control points of the Bézier triangles to obtain a $C^1$ surface. In Section V, it is for a selected case shown that without loss in accuracy, we obtain a $C^1$ surface satisfying all the above desired properties and with significantly lower curvatures than the two other main approaches. Section VI concludes the work.

II. RELATED WORK

Before we review the substantial amount of work on creating $C^1$ surfaces from Bézier triangles, it is important to emphasize a distinction, namely the difference between $C^1$ and $G^1$ (first-order geometric continuity) across boundaries. The $G^1$ continuity means that the first derivatives are not equal at the boundaries, but only pointing in the same directions so that their cross products are aligned, creating a visually smooth transition. This is acceptable for computer graphics, but is problematic when needed for trajectory generation.

Bézier triangles, referred to as Triangular (Bernstein-) Bézier patches in some literature, had great interest back in the 1970’s and 1980’s where explicit methods for local interpolation were derived for different degrees of smoothness, $C^r$ ($r$ times continuous differentiable). As it will be shown throughout this section, most of these methods have the drawback of needing information about vertex and edge derivatives for their creation.

In the local schemes each Bézier triangle or a pair of Bézier triangles is created separately using data from only neighboring triangles or triangles in a small area around the one(s) being created. One of the first described local interpolation method is the $C^0$ nine parameter interpolant [14] ensuring quadratic precision of the interpolant using cubic Bézier triangles (see Fig. 2(a)) requiring position and gradient data for each triangle vertex. The need for gradient data, i.e. $C^1$ data, while only obtaining a $C^0$ surface are obvious drawbacks of the method. However, we will show in section III that the constraints for $C^0$ are quite easily satisfied, and a $C^0$ surface can thus also be obtained from position and normal data at each triangle vertex [15], referred to as Curved Point-Normal triangles. The $C^1$ quintic interpolation method [14], [16] using quintic Bézier triangles requires up to second derivative data at the vertices while also requiring a cross-boundary derivative at each edge midpoint, but only produces a $C^1$ surface although the interpolant is quintic. The method requires 21 pieces of information and is thus referred to as the $Q_{21}$ method. However, through condensation of parameters the need for cross-boundary derivatives can be omitted, creating the condensed interpolant, $Q_{18}$, method. The former method extends to the general case in [14], which can create a $C^r$ surface, using Bézier triangles of degree $n \geq 4r + 1$ and known vertices derivatives of up to order $2r$.

To remove the need for second order derivatives at the vertices for a $C^1$ surface, a popular approach is to use the $C^1$ Clough-Tocher interpolant [17], [18], [19]. The interpolant
works by splitting each triangle, referred to as the macro-triangles into three 'mini-triangles' by joining each vertex to the centroid (see Fig. 2(b)). Each mini-triangle is then employed as a cubic Bézier triangle satisfying $C^1$ cross-boundary continuity (and $C^2$ in their common meeting point) from only 12 data: vertex position and gradients along with cross-boundary derivatives. Again the cross-boundary derivatives can be omitted using condensations of parameters. A modified version of the Clough-Tocher interpolant is given in [20], in which the cross-boundary derivative condition is changed from being linear to try and come as close to $C^2$ as possible. A range restricted version of the standard Clough-Tocher scheme is found in [21]. A similar approach is the $C^1$ Powell-Sabin interpolant [22], which requires the same 12 data, but instead splits the macro-triangles into either 6 or 12 mini-triangles depending on the geometry of the macro-triangle. The mini-triangles are also only of quadratic degree.

The case of obtaining a $C^r$ surface knowing only $C^r$ data is not restricted to $r = \{1,2\}$ and the formulas describing $C^r$ continuity along with a generalization of the $C^1$ Clough-Tocher scheme to the $C^2$ case is given in [23], while an explicit bivariate $C^2$ Clough-Tocher scheme using position, gradient and Hessian at the vertices is given by [24]. For our purpose, $C^r$ methods for large $r$ are inadequate as they require high derivatives at the vertices. For example, in [25], 9th degree Bézier triangles are created explicitly through a condensed scheme, but requires fourth order derivatives at the vertices.

An alternative to subdivision is the use of convex combinations. In these approaches, multiple sets of internal control points are used to obtain the desired continuity for the respective edges. Examples of this include the original version of the scheme [26] where $C^1$ is obtained using first derivatives for cubic Bézier triangles with three versions of the internal control (see Fig. 2(c)). The scheme also exists for higher continuity constraints, e.g. [27] which uses quintic Bézier triangles to obtain $C^2$ with second order derivatives, while it is also seen in applications with range and convexity restrictions [28], [29], [30], [31], [32]. In [33] a shape operator is included in the representation, while a new method uses a quartic Bézier triangle to obtain $C^1$ that only needs 10 control points as in a cubic [34] albeit still needing first derivatives at the vertices.

Global schemes will either use global data only available by considering the whole surface to create each separate Bézier triangle or create the whole surface in one go by solving a global system of equations. An approach for range restricted $G^1$ and $C^1$ interpolation based on a global scheme is presented in [35]. In this, quartic Bézier triangles are used and partially created knowing vertex positions and estimated derivatives. The authors then mention the possibility to find the remaining control points using a least squares approach to obtain $G^2$ or $C^1$, but deems this approach flawed as it might produce over-flatness. Instead they propose to create a global optimization problem using minimized sum of squares of principle curvatures with respect to the $G^1$ constraints. However, it is unclear if they use a similar approach for $C^1$.

Similar approaches as above and variations of these are also seen in the area of computer graphics where the $C^r$ continuity is not necessarily required, but rather only producing approximate $C^1$ [36], $G^1$ [37], [38], [39] or $G^2$ [40]. This is also found in more complex areas using rational variations in for example analysis with Bézier tetrahedra [41] and isogeometric analysis for solving PDEs [42].

All the above schemes require derivative information of at least first order. For these methods, the derivatives are very important to estimate well as these strongly impact the shape of the final surface. However, it is also difficult to estimate derivatives when only the scattered position data from often very irregular triangulations are available. An overview of some estimation methods can be found in [43], while the method often used in the above literature is given in [44]. As it can also be seen in these papers, no method is truly better for all cases, and different methods and parameters might have to be used, depending on the tessellation and ordering of the scattered data points. It is thus highly desirable to avoid the need for accurate derivative information for the creation of the surfaces.
III. CUBIC BÉZIER TRIANGLE

As Bézier triangles are not so known, we briefly review them here. For more details see e.g. [14]. For a triangle mesh, \( M \subset \mathbb{R}^3 \), an arbitrary point, \( p \in M \), can be described as a linear combination using the barycentric coordinates of the point, \( \{ \gamma_0, \gamma_1, \gamma_2 \} \), with respect to the vertices, \( V_0, V_1, V_2 \), of the triangle, \( T \in \mathbb{R}^3 \), in which the point resides:

\[
p = \gamma_0 V_0 + \gamma_1 V_1 + \gamma_2 V_2, \quad \gamma_0, \gamma_1, \gamma_2 \geq 0, \quad \gamma_0 + \gamma_1 + \gamma_2 = 1 \tag{1}\]

This definition gives a linear interpolated precision of the mesh, which is only \( C^0 \) at the boundaries between the triangles. Using a Cubic Bézier triangle, one can obtain a cubic interpolated precision:

\[
P(u,v,w) = \sum_{i+j+k=n, \ i,j,k \geq 0} b_{i,j,k} \frac{n!}{i!j!k!} u^i v^j w^k, \quad u+v+w=1, \quad n=3 \tag{2}\]

\[
= u^3 b_{300} + 3u^2 v b_{210} + 3u^2 w b_{201} + 6uvw b_{120} + v^3 b_{030} + 3v^2 w b_{021} + 6uvw b_{112} + w^3 b_{003}
\]

where \( b_{i,j,k} \in \mathbb{R}^3 \) are the 10 Bézier ordinates as shown in Fig. 3. Notice that we use the convention \( 0^0 = 1 \).

From this definition it is easy to see that as long as \( b_{300} = V_0, b_{030} = V_1, b_{003} = V_2 \), the vertices of the triangle in the interpolation are included at the coordinates \( u = 1, v = 1 \) and \( w = 1 \), respectively. However, this constraint is not enough to ensure even \( C^0 \) across the boundaries. As the boundaries of the Bézier triangle are in fact (univariate) Bézier polynomials [45], this means that for \( C^0 \) between two adjacent cubic Bézier triangles, the adjacent Bézier ordinates must be equal. For two adjacent triangles, \( \tau = \{ V_0, V_1, V_2 \} \) and \( \tau' = \{ V_3, V_2, V_1 \} \) with Bézier ordinates, \( b_{i,j,k} \) and \( \hat{b}_{i,j,k} \), respectively, this can be described as:

\[
\hat{b}_{0,0,k} = b_{0,k,0} \tag{3}\]

For the cubic case (see Fig. 4) this implies:

\[
\hat{b}_{030} = b_{003}, \quad \hat{b}_{021} = b_{012}, \quad \hat{b}_{012} = b_{021}, \quad \hat{b}_{003} = b_{030} \tag{4}\]

Opposed to most common methods, we do not need to use (difficult to estimate) gradient information at the vertices to create a \( C^1 \) surface directly. Instead, a \( C^0 \) initial surface using only vertex positions and triangle face normals at the vertices, is created. This initial surface is obtained using Curved PN (Point-Normal) Triangles [15], which reveals a \( C^0 \) surface as the constraints in (4) are satisfied. For estimation of the normal vectors at the vertices, the following very robust and intuitive formula given in [46] is utilized:

\[
N_{V_i} = \sum_{i=1}^{n} \alpha_i N_T_i \tag{5}\]

where \( N_{V_i} \) is the direction of the normal vector, the summation is over the triangles with \( V_i \) as a vertex, \( \alpha_i \) is the angle between the two edges of triangle \( T_i \) connected to the vertex \( V_i \), and \( N_{T_i} \) is the face normal of that triangle.

IV. INCREASING SURFACE TO \( C^1 \)

In contrary to the local interpolation schemes which requires either higher derivatives, subdivision or convex combinations, global interpolation schemes does not necessary require any of this. In [47], it is for a general triangulation described how it is possible to obtain \( C^1 \) across the boundaries by solving a large sparse linear system over the whole surface. In this section, this scheme is explored for Cubic Bézier triangles, as these provide an intuitive representation for manipulation of the surface, while the boundaries are also (univariate) Bézier curves. The ordinates from the created \( C^0 \) surface in the previous section are used as the initial values in the linear system of equations, making it a minimization problem for satisfying the constraints, while staying as close to the initial surface as possible.

This is somewhat similar to the least squares approach mentioned in [35]. However, as we use lower degree polynomials and initial values based on the estimated normal vectors, we avoid undesired "flattening" and hence better capture the real surface.

A. Prerequisites for \( C^1 \) between Cubic Bézier Triangles

Extending the set of constraints to obtain \( C^0 \) between two adjacent triangles given in (4), the continuity can be raised [14] to \( C^1 \) if the following additional constraints [48] are satisfied:

\[
\hat{b}_{1,j,k} = \gamma_0 b_{1,k,j} + \gamma_1 b_{0,k+1,j} + \gamma_2 b_{0,k,j+1} \tag{6}\]

where \( \{ \gamma_0, \gamma_1, \gamma_2 \} \) are the barycentric coordinates of \( V_3 \) given in triangle \( \tau = \{ V_0, V_1, V_2 \} \) (see Fig. 4).

For the cubic case this implies these additional constraints to those of (4):

\[
\hat{b}_{1,0,k} = \gamma_0 b_{1,0,k} + \gamma_1 b_{0,k+1} + \gamma_2 b_{0,k+1} = b_{1,0,k} \tag{7}\]

If the constraints from (3) and (6) are similarly applied to the two remaining sides of the triangle, it is clear that all ordinates except \( b_{111} \) are part of the constraints for two sides, while \( b_{111} \) is for all three sides. This is the reason these must be solved as a global problem unless a local subdivision or

![Fig. 3. Bézier ordinates of a Cubic Bézier Triangle patch.](image-url)
convex combinations of three different $b_{111}$’s are applied as discussed in the related work section.

**B. Parametrization through Harmonic Mapping**

The papers discussed in the related work section are divided into a resulting surface either on the form $z = f(x,y)$ or on the parametrized form $\{x(u,v), y(u,v), z(u,v)\}$. In this paper, we choose the form $\{x(u,v), y(u,v), z(u,v)\}$, as the work pieces may bend so much that they contain multiple $z$-values for the same $(x,y)$ coordinate. This also means that the Bézier ordinates in (2) will be three dimensional instead of one dimensional.

An advantage with the $z = f(x,y)$ formulation is that for a triangle vertex $(x_i, y_i, z_i)$, the parameters $(x_i, y_i)$ are directly available, whereas for the parametrized form, the parameters $(u_i, v_i)$ must be found. There are several algorithms for this problem. Here, we use a Harmonic Mapping [49] that creates a piecewise linear mapping for the triangulation, which is confined inside a convex polygon in $\mathbb{R}^2$. The polygon degree is determined by a number of boundary vertices that the user selects, and are placed proportional to the arc length between the vertices on a circle. The map can be interpreted as the distortion energy of a configuration of springs, where each edge in the triangulation is modelled as a spring. The piecewise linear map is then given as the state of this system that minimizes the total metric distortion energy. This state can be obtained by solving a sparse linear least squares problem.

For the example used in Section V, a triangulation with 554 triangles as shown in Fig. 1, the Harmonic Map with the choice of 4 boundary vertices yields the $(u,v)$ parameters for the triangles shown in Fig. 5.

**C. Continuity Constraint**

Using the parametrization in Fig. 5, the constraints for $C^1$ given in (7) can be written as a system of equations over the whole surface. We split this process into two parts: around vertices in a counter-clockwise direction and between adjacent triangles. Referring to Fig. 4, the constraint around $V_2$ for the shown two triangles with ordinates that are initialized using PN triangles as discussed above will be:

$$\gamma_0 r_0 + \gamma_1 r_1 - r_3 = \gamma_0 \beta_{102} - \gamma_1 \beta_{012} - \gamma_2 \beta_{003}$$

where $r_0, r_1, r_3$ correspond to the unknown offsets needed to be applied to $\beta_{102}, \beta_{012}, \beta_{120}$, respectively, to obtain $C^1$. We emphasize that no offset is applied to $\{\beta_{003}, \beta_{030}\}$ as it was required for the method to include the original vertices of the triangles. The same procedure is applied around every vertex which are connected to more than one triangle. If the vertex is inside the triangulation, $n - 1$ constraints is needed, while boundary vertices will need $n - 2$ constraints, where $n$ is the number of triangles connecting to that vertex. Notice that each constraint is 3 dimensional, containing the value and offset $r = (r_x, r_y, r_z)$, likewise for the Bézier ordinates.

The second set of constraints is given between each pair of triangles, again referring to Fig. 4:

$$\gamma_0 r_0 + \gamma_1 r_1 + \gamma_2 r_2 - r_3 = \hat{\beta}_{111} - \gamma_0 \beta_{111} - \gamma_1 \beta_{021} - \gamma_2 \beta_{012}$$

where $r_o, r_1, r_2, r_3$ here correspond to the unknown offset for $\beta_{111}, \{\beta_{021}, \hat{\beta}_{012}\}, \{\beta_{012}, \beta_{021}\}, \hat{\beta}_{111}$, respectively. Here all four points can be modified. The same procedure is applied between each pair of triangles.

The obtained number of 3D vector constraints and 3D vector unknowns agrees with the number of scalar constraints and unknowns from the formulation given by [47]:

$$Unknowns = 10m_2 + 20m_1 - 20$$

$$Constraints = 8m_2 + 19m_1 - 21$$
where \( m_1 \) is the number of inner vertices and \( m_2 \) the number of boundary vertices. As can be seen, the number of constraints is always larger than the number of unknowns. Moreover, there are 2 extra variables per boundary edge, which means that the complete boundary can be constrained to fit to a prescribed cubic Bézier representation of the boundary of the surface. Even with fixing the boundaries in this way, the resulting large sparse linear system of equations can then be minimized used a least squares approach, as the system has more unknowns than constraints.

V. RESULTS

In this section, the quality of the Bézier triangle surface produced by the global method presented in this paper is examined with respect to the suitability for machining. We do this by comparing our method with two local cubic \( C^1 \) interpolation methods based on convex combination and Clough-Tocher respectively. The focus is purely on the quality of the methods and not on computational speed as the focus is on offline trajectory generation where all methods by far are fast enough. We selected the convex combination method from [26] in which the term for the single center ordinate in (2), \( \cdots + 6uvw_{111} + \cdots \), is replaced with the convex combination of the three center ordinates, \( \cdots + 6uvw v^2u^2v_{111} + v^2u^2v_{111} + v^2u^2v_{111} + \cdots \), including proper computational handling of the removable singularities at the corners. The Clough-Tocher split method representative was created by the three different methods. As can be seen in Fig. 7, the Clough-Tocher method introduces larger spikes which is due to that each triangle is split into three smaller triangles. For machining, it is often desirable to go with a constant machining speed \( v \). This is achieved by introducing a parameter scaling \( s(t) \) where \( t \) is time, so that \( s'(t) \equiv \|r'(s)\| \). There are no severe spikes in the \( \|r'(s)\| \) curves even for Clough-Tocher, so this time scaling is easily applicable with all three methods.

For machining curves that are executed with constant speed, the acceleration is proportional to the curvature \( \kappa(s) = \frac{\|r'(s) \times r''(s)\|}{\|r'(s)\|^3} \). Hence, introduction of high curvatures in the interpolation is very undesired. It should be noticed that curvatures are independent of the parameter scaling \( s'(t) \) and hence curvature is an excellent measure for the suitability for machining. In Fig. 8, we show the curvatures for each of the three methods along with the curvature of the original surface. From these plots, we can clearly see that the presented global method have curvature spikes that are not much larger than the curvatures from the original surface, whereas both Clough-Tocher and the convex combination method introduces very large spikes. Hence, our method is significantly more suitable for machining than two prominent methods from the state-of-the-art.

VI. CONCLUSION

In this paper, a new method for generating \( C^1 \) surfaces from triangle meshes has been presented. The method employs cubic Bézier triangles over the triangle mesh, retaining the triangular form and increases the continuity to \( C^1 \) by minimizing the surface perturbation, subject to the large

![Fig. 6. Triangle Bézier surface created using presented global method.](image-url)
sparse system of constraint equations, while maintaining the original vertex positions. The main benefit of the method is that it generates surfaces with much lower curvature spikes than two prominent examples of existing methods, while maintaining similar positional accuracy. It should also be mentioned, that contrary to almost all existing methods, it is not necessary to specify vertex derivatives where there are no really robust algorithms for estimating these. Hence, it is easier to apply and with the lower curvatures, undesirable interpolation artifacts that would reduce robot machining quality are avoided.

### Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Error $\Delta p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Method</td>
<td>$0.013655 ± 0.00001$</td>
</tr>
<tr>
<td>(Described above)</td>
<td></td>
</tr>
<tr>
<td>Convex Combination</td>
<td>$0.013651 ± 0.00001$</td>
</tr>
<tr>
<td>[26]</td>
<td></td>
</tr>
<tr>
<td>Clough-Tocher</td>
<td>$0.014650 ± 0.00001$</td>
</tr>
<tr>
<td>Variation as in [20]</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8. Curvature, $\kappa = \frac{|r'(s) \times r''(s)|}{|r'(s)|^3}$, along three lines for the three different methods.

### References


