ISOMA-039.1

Automated Tolerance Specification Across Sensing, Design, and Manufacturing

Tarek M. Sobh, Thomas C. Henderson, and Frédéric Zana

Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, CT 06601, USA

Abstract

In this work we address the problem of tolerance representation and analysis across the domains of industrial inspection using sensed data, CAD design, and manufacturing. Instead of using geometric primitives in CAD models to define and represent tolerances, we propose the use of stronger methods that are completely based on the manufacturing knowledge for the objects to be inspected. We guide our sensing strategies based on the manufacturing process plans for the parts that are to be inspected and define, compute, and analyze the tolerances of the parts based on the uncertainty in the sensed data along the different toolpaths of the sensed part.

Keywords: Tolerance, Manufacturing, Sensing, Inspection, Industrial Design

1 Introduction

In this work we address the problem of recovering manufacturing tolerances and deformations from the uncertainty in sensing machine parts. In particular, we utilize the sensor uncertain y to recover robust models of machine parts, based on the probabilistic measurements recovered, for inspection applications. We design and implement a spline-based model that captures manufacturing tolerancing based on uncertain sensed data and knowledge of possible manufacturing process plans.

2 Background, Motivation, and Methodology

We propose toolpaths with tolerances as an instance of the manufacturing process (process plan) that provides a unifying approach to dealing with tolerance and sensing issues across design, manufacturing and inspection. The use of interval Bezier curves for a complete description of approximation errors was proposed by Sederberg and Farouki[5] (see paper for details). The basic idea is to extend splines to polynomials whose coefficients are intervals with well defined arithmetic operations. Such splines define a region in space rather than a curve. This not ion captures very nicely the semantics of a tolerance specification. We have developed interval curves for both 2D and 3D and algorithms based on interval splines for machine toolpath representation.

Given the CAD geometry for a part, a tolerance specification, and a class of NC mill to be used, then generic knowledge about such mills can be used to generate a desired toolpath with its associated tolerance (call it TP_d. Once a specific mill is selected, then the nominal toolpath from TP_p together with the accuracy of the mill determine the actual toolpath (call this TP_a These two toolpaths allow us to determine a great deal about the efficiency and uncertainty regions of the process. First, if $TP_a \subset TP_d$ is true, then we

know that the tolerance should, in principle, reachieved. If $TP_d - TP_a$ is large, then the selected machine may be too precise, and therefore, too expensive. If the boundary of TP_a *is* close to that of TP_d, this signals places where sensing may be necessary to guarantee the inclusion relation. This also gives insight into how accurate the sensing needs to be. Even if TP_a is not contained in TP_d, this approach allows us to estimate what percentage of milled parts will be out of spec, and thus an informed decision can be made whether to tighten the accuracy of the machine, or where to sense with high probability of part error. Thus, the toolpath representation allows insight into design, manufacture and inspection in a common framework.

3 Interval Splines and Generalization: Checking that all points reach the tolerance goal

3.1 Interval Splines

The use of interval Bézier curves for *a* complete description of approximation errors was proposed by Sederberg and Farouki[5]. The basic idea is to extend splines to polynomials whose coefficients are intervals with well defined arithmetic operations. Such splines define a region in space rather than a curve. This not ion captures very nicely the semantics of a tolerance specification, especially when it is generalized in 3D: if the assumption is made that the sensing error is Gaussian, then it can be described it by an ellipsoid around each sensed point (using a step value). Thus, along a sensed toolpath, an offset surface is produced (see [3]). We have only assumed that the enclosing envelopes are described by ellipses in planes orthogonal to the toolpath. Hence our algorithm allows for representing volumetric error and can easily be extended to other shapes than ellipses - which means different offset surfaces. This approach will require the ability to answer the question: is one ellipse inside the other one ? as fast as possible - when they are in the same plane. The final test will be to check the reliability of the proposed algorithm on real sensed data, along manufacturing toolpaths on parts that are inspected.

The algorithm uses a property that is associated with curves of the same degree, which is the basis of interval splines. Since a Bézier curves of degree k is deduced from the control point by the recursive equation (see [4]):

$$\begin{cases} P:(t) = P_i \quad (j-k \le i \le j) \\ \quad \text{and for } 0 \le r \le k-1 \\ P_i^{r+1}(t) = W;(t) + (1-t)P_{i-1}^{r}(t) \\ \quad \text{when } j-k+r \le i \le j \\ P_j^k(t) = s(t). \end{cases}$$

For curves of same degree, if the corresponding control points are on a line (resp. on a plane), then during this recursive process each corresponding p:(t) will also be on a line (resp. on a plane), hence for all t the different evaluations $(S_1(t), S_2(t) \ldots)$ will give points on a line (resp. on a plane). An easy way to ensure that the control points are on a line is to have initial points on a line too, since the control points are deduced by a linear operator.

3.1.1 2D Interval Splines

In our 2D representation, an interval is a set of 3 points (corresponding to the nominal point and two bounds). The spline interpolation is done (on 6 consecutive points) separately on each of the 3 corresponding curves (see Figure 1). Note that evaluation at any parameter $t \in [0, 1]$ yields 3 points on a line.

As indicated above, the determination of inclusion of one interval spline within another is important. Figure 2 shows the case where inclusion is true.

We have developed a technique to answer this question (see section 3.2.2). Moreover, if one interval contains another, then the 2-D difference of the two intervals is also possible to determine.

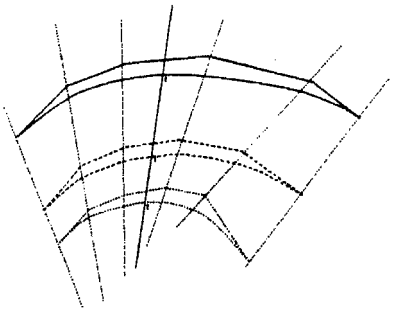

Figure 1: One Interval Spline

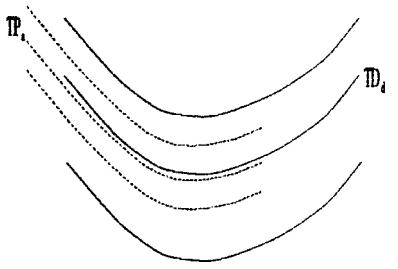

Figure 2: $TP_a \ C \ TD_d$

3.1.2 3D Interval Splines

In 3D, we've assumed that the uncertainty around a point is described by an ellipse (in the plane normal to the curve). Thus, we also use 3 points to describe the ellipse (X for the nominal point, and X_1 and X_2 the two extreme points along the two axis of the ellipse). The problem reduces to determining whether one ellipse is inside another. We have developed an algebraic solution to this problem (see section 3.2.3).

3.2 Description of the Algorithm

There is no significant difference between the 2D and the 3D algorithm, except for the part that compares two intervals (resp. two ellipses). Both algorithms use a procedure to check if the interval spline from the sensing device (We used a GRF-2 light stripper scanner) is inside the interval spline of the allowable tolerance model.

3.2.1 Common part

To verify that one interval spline is inside another, the following three steps are used:

1^{st}: Putting the parameters of the 2 splines together:
We want to ensure that for all t the two corresponding intervals are on the same line (resp. in the same plane, for ellipses). We implement a divide and conquer algorithm, using the sign of:

$$\det \begin{vmatrix} x(t) & x_1 & x_2 \\ y(t) & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}$$

or (in the 3D case)

$$\det \begin{vmatrix} x(t) & x_1 & x_2 & x_3 \\ y(t) & y_1 & y_2 & y_3 \\ z(t) & z_1 & z_2 & 23 \\ 1 & 1 & 1 & 1 \end{vmatrix}$$

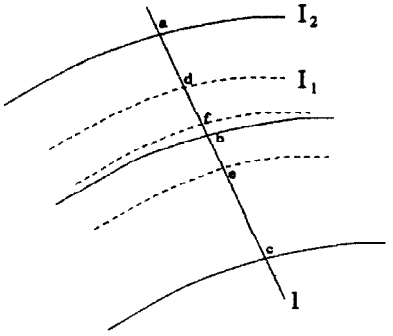

Figure 3: Included Interval Spline

Those two determinants are the equations of the lines (or the plane where the ellipse lies) that correspond to one interval spline, thus the algorithm cuts the second interval spline to redefine it (the determinant utilizes the initial points used to define the first interval spline at the beginning). So there is no need to have two interval splines of same degree at the beginning, since the second one is completely rebuilt (with the same degree, and control points on the same line or plane as the first interval spline). See figure 3 where l = (a, *b, c)* cuts the interval spline I_1 in *d, f* and e to define a new interval: with classical methods, that have to be done (see [6]).

2^{nd}: Compare as many intervals as possible.
Now that the intervals came together, this part is computable in O(n) where n is the number of points on a spline (resp. ellipses).

3^{rd} When 2^{nd} fails, check if it's an ending
If not, then the inclusion fails. This check has to be made as both splines do not necessarily begin or end at the same time.
To check an ending, the methods in 2D and 3D are very similar. The method utilizes the fact that the sign of the determinant of vectors gives the orientation of such a frame - when it is compared to the canonic frame. Hence, comparing two determinants can decide whether two points are on the same side of a line or a plane. See figure 4 for the 2D vectors.

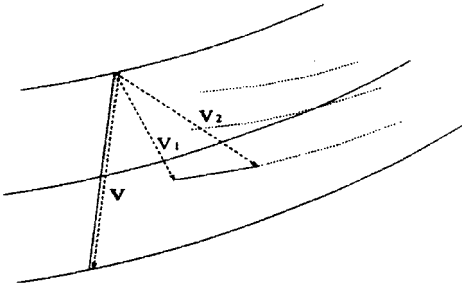

Figure 4: Two Interval Splines

For example, in 2D the signs of $det(V, V_1)$ and $det(V, V_2)$ are compared. A same sign means the points are on the same side.

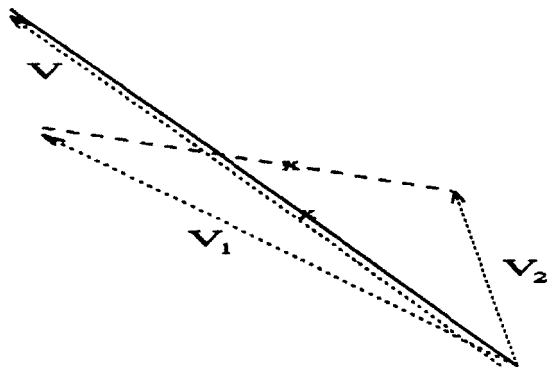

Figure 5: How to compare two intervals that are not necessarily parallel

3.2.2 Comparing two intervals

Here we check to ensure that O < $V.V_i < \|V\|^2$ (i = 1, 2), and to check the angles between the vectors (V, V_i) (i = 1,2) (see figure 5).

3.2.3 Algebraic Solution to Ellipse Inclusion

If the two ellipses do not intersect and if the center of one is inside the other, then one is contained by the other one. For the intersection of ellipses, we have developed an algebraic solution using the St urm Theorem (see [1] or [2] for more details).
We assume that the implicit equation of the ellipse with center X, and which go through the extreme points X_1 and X_2 (assumed to be along the 2 orthogonal axis, but it is not necessarily the case along the curve) is given by the following:

$$\text{take } \vec{V_1} = \frac{(X_1 - X)}{\|X_1 - X\|^2} \text{ and } \vec{V_2} = \frac{(X_2 - X)}{\|X_2 - X\|^2} \text{ then:}$$

$$M \in ellipse \iff (\vec{XM}.\vec{V_1})^2 + (\vec{XM}.\vec{V_2})^2 = 1$$

We also also assume that the second ellipse has the following parametric equation (same approximation):

$$M(t) = x' + \frac{2t}{1+t^2}\vec{X_1'X'} + \frac{(1 - t^2)}{1+t^2}\vec{X_2'X'}$$

substituting this point in the implicit equation of the other ellipse gives the following polynomial of degree 4:

$$(\vec{XX'}.\vec{V_1} + 2t\vec{X_1'X'}.\vec{V_1} + (1 - t^2)\vec{X_2'X'}.\vec{V_1})^2$$

$$+(\vec{XX'}.\vec{V_2} + 2t\vec{X_1'X'}.\vec{V_2} + (1 - t^2)\vec{X_2'X'}.\vec{V_2})^2 = (1 + t^2)^2$$

The real roots - if they exist – realizes up to 4 points of intersection of those 2 ellipses, The Sturm theorem on polynomials suggests an algorithm to find the number of roots of any polynomial. If this algorithm is applied on a polynomial with symbolic variables as its coefficients, one can get a condition that determines when (and only when) the polynomial has a real root. If this is performed on the polynomial $X^4 + aX^2 + bX$ + c we find[1]:

$$\Gamma = 2a^3 - 8ac + 9b^2$$
$$A = 16a^4c - 4a^3b^2 - 128a^2c^2 + 144ab^2c - 27b^4 + 256c^3$$

*result taken from the course "geométrie semie-algébrique" from Professor Coste (Universit y of Rennes, France), DEA IMA.

$X^4 + aX^2 + bX + c$ has no real roots if and only if
$(a \geq 0 \text{ and } \Delta > 0)$ or $(a > 0 \text{ and } \Gamma = 0)$ or $(a < 0 \text{ and } \Gamma > 0 \text{ and } \Delta > 0)$

If the polynomial $X^4 + dX^3$ is viewed as the beginning of the expansion of $(X + \alpha)^4$ then one can see that an appropriate translation transforms any degree 4 polynomial into a polynomial $T^4 + aT^2 + bT + c$ with $T = X - \alpha$. For our problem, the resulting values of a,b and c are given by the equations:

$$A1 = -X_2'\vec{X}'.\vec{V}_1 \qquad B_1 = 2X_1'\vec{X}'.\vec{V}_1$$
$$A_2 = -X_2'\vec{X}'.\vec{V}_2 \qquad B_2 = 2X_1'\vec{X}'.\vec{V}_2$$
$$C_1 = (X\vec{X}' + X_2'\vec{X}').\vec{V}_1 \quad C_2 = (\text{xi}' + X_2'\vec{X}').\vec{V}_2$$
$$A = \sqrt{A_1^2 + A_2^2} \qquad B = \sqrt{B_1^2 + B_2^2}$$
$$C = \sqrt{C_1^2 + C_2^2}$$

then $P(t) = c_4t^4 + c_3t^3 + c_2t^2 + c_1t + co$ with

$$c_4 = A^2 - 1 \qquad c_3 = 2(A_1B_1 + A_2B_2)$$
$$c_2 = B^2 + 2(A_1C_1 + A_2C_2 - 1)$$
$$c_1 = 2(B_1C_1 + B_2C_2) \qquad c_0 = C^2 - 1$$

and finally, we can find α and then a,b and c:

$$\alpha = \frac{c_3}{4c_4} \qquad a = \frac{c_2 - 6c_4\alpha^2}{c_4}$$

$$b = \frac{c_1 - 4c_4\alpha^3 - 2\alpha(c_2 - 6c_4\alpha^2)}{c_4}$$

$$c = \frac{\text{co} - c_4\alpha^4 + \alpha^2(c_2 - 6c_4\alpha^2) - \alpha(c_1 - 4c_4\alpha^3)}{c_4}$$

4 Experimental Results

The algorithm was tried on real sensed data, from the GRF-2 scanner, along a toolpath from a manufactured cover plate pocket. Figure 6 shows the part under inspection. Figure 7 includes range data from the scanner for the pocket in the cover plate. Figure 8 shows a CAD model for the pocket.

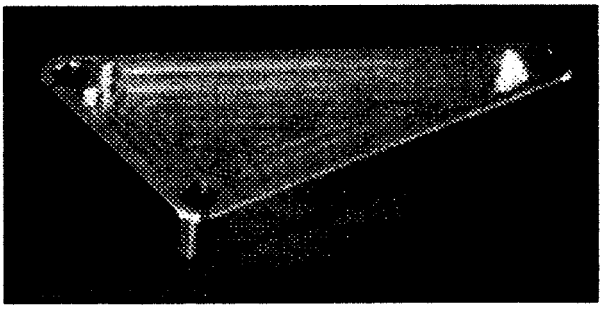

Figure 6: The machine part under inspection

The scanner was not very accurate, so first we recognized pieces of lines and arcs out of the noisy points from the scanner and defined those as our nominal curve. This is not a bad approximation, as the NC milling machine tool actually moves only in *straight line* and *curve* segments. For each points from the scanner we find the closest point to this nominal curve and - eventually - increase the radius of the sphere around the nominal point to

Figure 7: Range data for the pocket

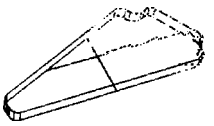

Figure 8: A CAD model for the pocket

include the point from the scanner. Finally, we smooth the values from the radius 40 times and define the surface with circles orthogonal to the path. Our algorithm compares it to the tolerance spline model, a few runs produced a good idea of the minimum specifications. Notice that both nominal curves from the model and from the scanner are quite different at some spatial instances, certainly because of a scale factor or a deformation from the scanner. Accurate data from a CMM along a toolpath would produce a much more precise input for the algorithm.

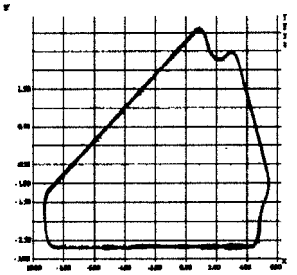

Figure 9: The points from the scanner and the computed offset surface cutting of the inner pocket

The figures 9 and 10 represent the inner profile, and figure 11 is the outer profile of the cover plate pocket. For the first one, we have found that the specification tolerance radius around the nominal curve of the model should be more than 0.12 cm. For the outer profile, we have found that it should be more than 0.065 cm. If the design tolerances do not meet the above requirements, then, *based on the sensed data*, the part should be rejected as it does not meet the required tolerances. It should be obvious that more precise results can be obtained with more runs. As one can see on the cross section of the outer pocket (figure 11), a few bad points can badly influence the result, specially if there is already an error between the two nominal curves.

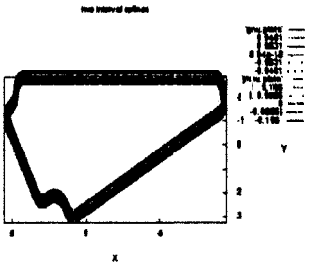

Figure 10: The 3D offset surfaces to compare

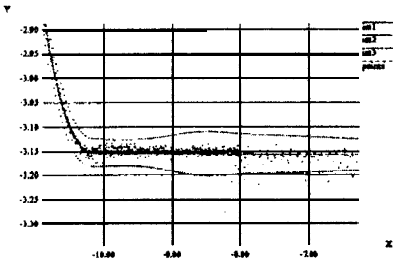

Figure 11: The points from the scanner and the computed offset surface: a detail of the outer pocket

5 Conclusions

We propose toolpaths with tolerances as a unifying approach to dealing with tolerance issues across design, manufacturing and inspection. Not only does this permit us to answer quest ions concerning design and manufacturing processes, but also gives a way to determine places in the process and on the part where sensing is useful to ensuring that tolerances are met. We have developed algorithms and implemental ions based on interval splines.

References

[1] ARNON, MIGNOTTE, AND LAZARD. In Algorithms *in real algebraic geometry* (1988), Arnon and Buchberger, Eds., Academic press.

[2] BENEDETTI, R., AND RISLER, J.-J. In *Real Algebraic and Semi-algebraic Sets (1990),* Hermann, pp. 8–19.

[3] BRECHNER, E. L. General offset curves and surfaces. In *Geometry Processing for Design and Manufacturing (1992),* SIAM, pp. 101-121.

[4] RISLER, J.-J. In *Méthodes mathématiques pour la CA O* (1991), Arnon and Buchberger, Eds., Masson.

[5] SEDERBERG, T. W., AND FAROUKI, R. T. Approximation by interval bézier curves. In *Computer Graphics and Applications* (September 1992), IEEE, pp. 87-95.

[6] WANG, K. Y. Parametric surface intersection. In *Geometry Processing for Design and Manufacturing* (1992), SIAM, pp. 187-204.