Journal of Intelligent and Robotic Systems **24:** 387–401, 1999.

Tolerance Representation and Analysis in Industrial Inspection

TAREK M. SOBH, THOMAS C. HENDERSON and FRÉDÉRIC ZANA
Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06601, USA

(Received: 17 June 1998; accepted: 25 July 1998)

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 tolerance of the parts based on the uncertainty in the sensed data along the different toolpaths of the sensed part. We believe that our new approach is the best way to unify tolerances across sensing, CAD, and CAM, as it captures the manufacturing knowledge of the parts to be inspected, as opposed to just CAD geometric representations.*

Key words: tolerance, manufacturing, sensing, CAD, CAM

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 uncertainty 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.

We design and implement our sensing strategies and tolerance determination algorithms based on interval splines. We believe this is the best way to define a unifying framework, as it captures both parameterizable manufacturing tolerancing errors, and non-easily-parameterizable ones (toolpaths that produce a surface definition, for example). This method is also suitable for our purposes as our CAD modeler (The Alpha_1 system, designed at the University of Utah) is based on spline representation, and it is used to produce process plans and toolpaths for NC

* This work was supported in part by DARPA grant N00014-91-J-4123, NSF grant CDA 9024721, and a University of Utah Research Committee grant. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

milling machines to manufacture the actual parts from CAD models. Our toleranceing method captures the mechanical way in which the manufacturing tool moves and actually *makes* a feature, surface or curve in a manufacturing process.

The standard representations for Computer Aided Design include volumetric, boundary and CSG models. Current advanced modelers, can produce process plans for specific machines in order to manufacture the object. We believe that the process plan and associated information (e.g., the tool path, the tool to be used, its speed, etc.) provide a strong basis for analyzing the manufacturing and inspection steps with respect to tolerance.

A tolerance specification on the shape geometry must be transformed into the corresponding tolerance on the machining operation and vice versa. This in turn can be used to select an approapriate manufacturing process, given knowledge of the manufacturing accuracy of the process. This yields direct methods for deciding on sensing strategies both to monitor the manufacture of the part, as well as for post-manufacturing inspection. These sensing strategies are derived from an analysis of where the toolpath is most likely to deviate from the tolerance specification.

These must all be done as efficiently as possible; in particular, it must be:

- straightforward to choose the cheapest manufacturing process, to go as fast as possible on that machine,
- to make as few sensed measurements as possible, and
- to perform as little computation as possible.

The keys to our approach are:

- have/use knowledge about each feature and machining process for that feature, and
- exploit the tool path representation to guide analysis and sensing strategies.

The usual approach to validation is to simply measure the geometry resulting from the manufacturing process and compare it to the nominal geometry from the CAD model. We believe that a stronger approach, exploiting knowledge of the process plan and the particular manufacturing process, is possible, and that this approach permits the automatic synthesis of sensing strategies.

To achieve this requires a tolerance specification which:

- specifies design geometry tolerance as well as toolpath tolerances, and
- helps locate high payoff (i.e., maximal information gain) inspection regions.

We are working with the Alpha_1 Computer Aided Geometric Design system and exploiting its ability to generate process plans for 3- and 5-axis NC milling machines. For these machines, the process is a set of toolpaths with appropriate

tools, speeds, etc., specified. Thus, a sensing strategy is a set of sensing operations carried out at particularly high risk parts of the toolpath or places on the completed part.

2. Background, Motivation, and Methodology

The traditional approach to structuring sensing strategies and tolerance computation for the inspection of machine parts has been to utilize the sensed data (range, image, and/or touch) and the recovered geometries of the sensed objects for guiding the sensors to get more data and to do better fittings at the "relevant" or "uncertain" regions. We propose an approach that is based on the knowledge of the actual manufacturing process for the parts to be inspected, as opposed to only the sensed data points and the recovered geometric CAD model. Our approach utilizes the knowledge of the process plan and the subsequent toolpath of the milling machines and the errors, uncertainties, and tolerances associated with that process to achieve an optimal sensing strategy at the relevant regions, features, and manufacturing path on the parts to be inspected. We anticipate that this approach will not only permit us to answer questions 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 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. We give examples of tolerance-based techniques for manufacturing features and for inspection purposes. The relation between part error models and tolerance specifications is outlined. The initial design of a unified framework for manufacturing-based sensing strategies for manufacture and inspection is given; the key is to tag tolerances to the manufacturing process itself (e.g., we use the toolpath and tolerance for NC milling).

The importance of quantifying tolerance in the specification, design, manufacturing and inspection process is obvious. Unfortunately, adequate representations of tolerance do not exist which permit dialog between these various aspects of the manufacturing process. This lack is particularly acute in systems which tightly integrate all of the aspects of prototyping (i.e., manufacturing, design, and sensing for inspection). We use the tolerance specification in conjunction with knowledge of the manufacturing process plans to determine more optimized sensing strategies. We propose to avoid the use of weak methods (e.g., comparing nominal geometry to dense range data from the actual part), and to synthesize strong process monitoring and inspection strategies based on detailed knowledge of geometry, tolerance specification, manufacturing features and processes, and the sensors involved.

The use of interval Bézier 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 notion 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. We have also implemented toolpath-based algorithms for answering tolerance questions in inspection of parts, and for structuring coarse-to-fine sensing strategies based on tolerance regions around sensed data.

Our goal is to develop a methodology which helps to guarantee that the intended tolerance specification is met as efficiently as possible. There issue we address in our framework is to validate that the tolerances have been achieved in the actual part that is inspected. This process involves sensor measurements either during the manufacturing phase or post-manufacture inspection. To ensure that the tolerance has been met, sensors are used to:

- measure the manufacturing process (e.g., table position during NC milling),
- measure parameters of manufacturing features (e.g., use a coordinate measurement machine to obtain hole diameter), and
- measure points on the surface directly and analyze them.

Of course, sensor error/uncertainty must be accounted for.

In order to structure the analysis process, we focus here on NC milling, and use the toolpath as the basis upon which design and manufacturing tolerance and sensor measurements will be compared. Much as operational semantics allows the meaning of a high level program to be defined in terms of the particular architecture upon which it executes, so can the CAD specification of a part be defined in terms of the machining operations which produce its shape. 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, be achieved. 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 notion 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]):

$$\begin{cases} P_i^0(t) = P_i, & j-k \leqslant i \leqslant j, \\ P_i^{r+1}(t) = tP_i^r(t) + (1-t)P_{i-1}^r(t), & \text{for } 0 \leqslant r \leqslant k-1, \\ P_j^k(t) = S(t), & \text{when } j-k+r \leqslant i \leqslant j. \end{cases}$$

For curves of same degree, if the corresponding control points are on a line (respectively on a plane), then during this recursive process each corresponding $P_i^r(t)$ will also be on a line (respectively on a plane), hence for all t the different evaluations $(S_1(t), S_2(t), \ldots)$ will give points on a line (respectively 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 2D difference of the two intervals is also possible to determine.

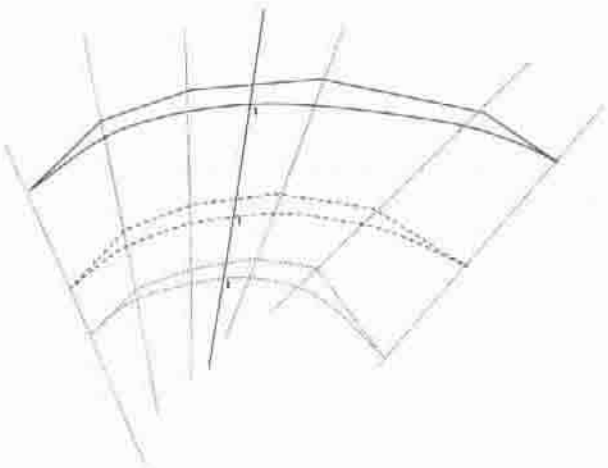

Figure 1. One interval spline.

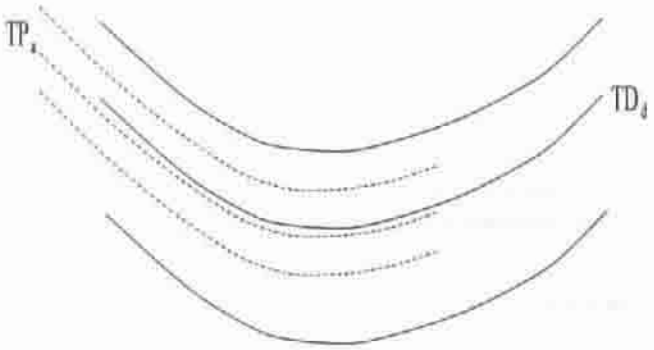

Figure 2. $TP_n \subset TP_d$.

3.1.2. *3D Interval Splines*

In 3D, we have 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 (respectively 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.

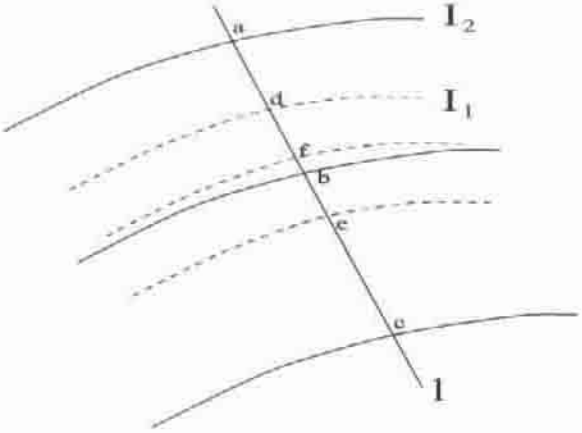

Figure 3. Included interval spline.

3.2.1. *Common Part*

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

Step 1. 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 (respectively in the same plane, for ellipses). We implement a divide and conquer algorithm, using the sign of:

$$\det \begin{vmatrix} x(t) & x_1 & x_2 \\ y(t) & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix} \quad \text{or (in the 3D case)} \quad \det \begin{vmatrix} x(t) & x_1 & x_2 & x_3 \\ y(t) & y_1 & y_2 & y_3 \\ z(t) & z_1 & z_2 & z_3 \\ 1 & 1 & 1 & 1 \end{vmatrix}.$$

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]).

Step 2. 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 (respectivelly ellipses).

Step 3. When step 2 fails, check if it is 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.

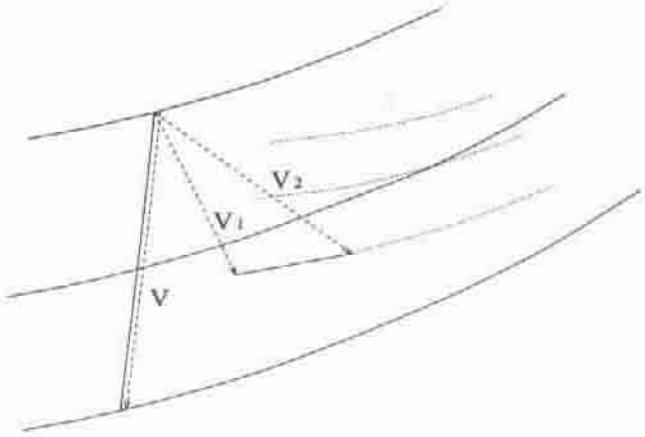

Figure 4. Two interval splines.

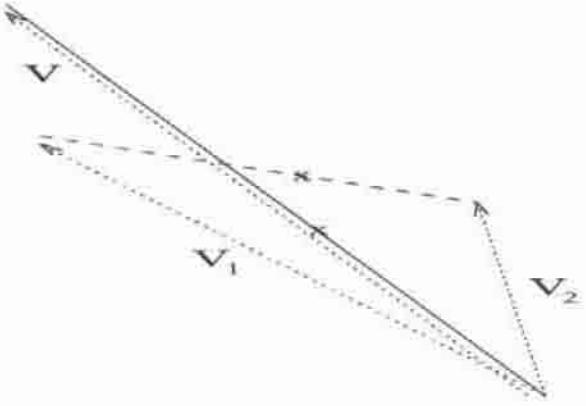

Figure 5. How to compare two intervals that are not necessarily parallel.

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.

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

3.2.2. *Comparing Two Intervals*

Here we check to ensure that $0 < V \cdot V_i < \|V\|^2$ $(i = 1, 2)$, 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 Sturm 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: take

$$\vec{V}_1 = \frac{(X_1 - X)}{\|X_1 - X\|^2} \quad \text{and} \quad \vec{V}_2 = \frac{(X_2 - X)}{\|X_2 - X\|^2}$$

then:

$$M \in \textit{ellipse} \iff \left(\overrightarrow{XM} \cdot \vec{V}_1\right)^2 + \left(\overrightarrow{XM} \cdot \vec{V}_2\right)^2 = 1.$$

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

$$M(t) = X' + \frac{2t}{1+t^2}\overrightarrow{X'_1X'} + \frac{(1-t^2)}{1+t^2}\overrightarrow{X'_2X'}$$

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

$$\left(\overrightarrow{XX'} \cdot \vec{V}_1 + 2t\overrightarrow{X'_1X'} \cdot \vec{V}_1 + (1-t^2)\overrightarrow{X'_2X'} \cdot \vec{V}_1\right)^2 + \left(\overrightarrow{XX'} \cdot \vec{V}_2 + 2t\overrightarrow{X'_1X'} \cdot \vec{V}_2 + (1-t^2)\overrightarrow{X'_2X'} \cdot \vec{V}_2\right)^2 = (1+t^2)^2.$$

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*:

$$\Gamma = 2a^3 - 8ac + 9b^2,$$
$$\Delta = 16a^4c - 4a^3b^2 - 128a^2c^2 + 144ab^2c - 27b^4 + 256c^3,$$
$X^4 + aX^2 + bX + c$ has no real roots if and only if
$(a \geqslant 0$ and $\Delta > 0)$ or $(a > 0$ and $\Gamma = 0)$ or $(a < 0$ and $\Gamma > 0$ 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

* Result taken from the course "geometric semic-algébrique" from Professor Coste (University of Rennes, France), DEA IMA.

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:

$$A_1 = -\overrightarrow{X_2'X'} \cdot \vec{V}_1, \qquad B_1 = 2\overrightarrow{X_1'X'} \cdot \vec{V}_1,$$
$$A_2 = -\overrightarrow{X_2'X'} \cdot \vec{V}_2, \qquad B_2 = 2\overrightarrow{X_1'X'} \cdot \vec{V}_2,$$
$$C_1 = \left(\overrightarrow{XX'} + \overrightarrow{X_2'X'}\right) \cdot \vec{V}_1, \quad C_2 = \left(\overrightarrow{XX'} + \overrightarrow{X_2'X'}\right) \cdot \vec{V}_2,$$
$$A = \sqrt{A_1^2 + A_2^2}, \qquad B = \sqrt{B_1^2 + B_2^2}, \qquad C = \sqrt{C_1^2 + C_2^2},$$

then $P(t) = c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0$ 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:

$$\alpha = \frac{c_3}{4c_4}, \quad a = \frac{c_2 - 6c_4\alpha^2}{c_4}, \qquad b = \frac{c_1 - 4c_4\alpha^3 - 2\alpha(c_2 - 6c_4\alpha^2)}{c_4},$$
$$c = \frac{c_0 - c_4\alpha^4 + \alpha^2(c_2 - 6c_4\alpha^2) - \alpha(c_1 - 4c_4\alpha^3)}{c_4}.$$

4. Experimental Results

4.1. TESTS ON SOME MATHEMATICAL CURVES

Tests were carried out both in 2D and 3D, but since 3D is more relevant to this project (and more difficult) we will only describe the 3D experiments. We have done some tests on 3D lines, parabolas, and *sin* curves, surrounded by ellipses that were allowed to turn around the central curve with different speeds. The tests show that it is very important to ensure that the surface do not cross itself, and that the algorithm will only compare the first connected component of the common part – thus, if there is an intersection only on the second connected component, the algorithm will not find it.

We have many results from different mathematical curves, and the algorithm works as expected, with or without an intersection. Figure 6 shows a case when the inside surface has been lifted enough to make an intersection. Figure 7 is a regular case.

4.2. TESTS ALONG THE SENSED TOOLPATH FOR AN INSPECTED COVER PLATE

The algorithm was tried on real sensed data, from the GRF-2 scanner, along a toolpath from a manufactured cover plate pocket. Figure 8 shows the part under

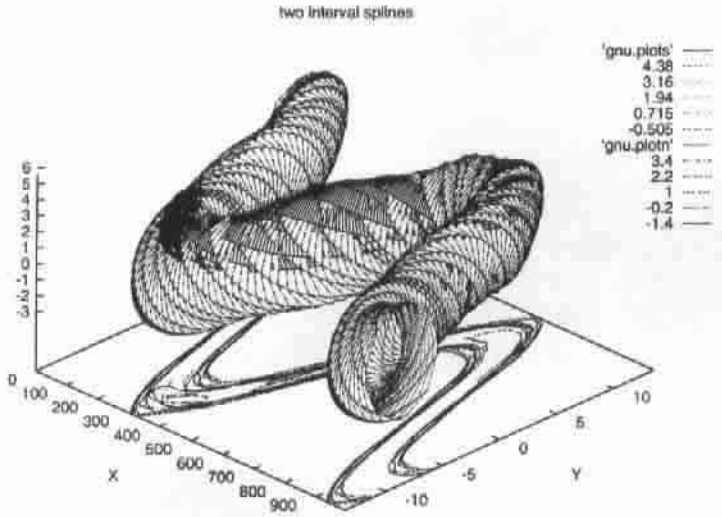

Figure 6. Case when the tolerance goal fails clearly.

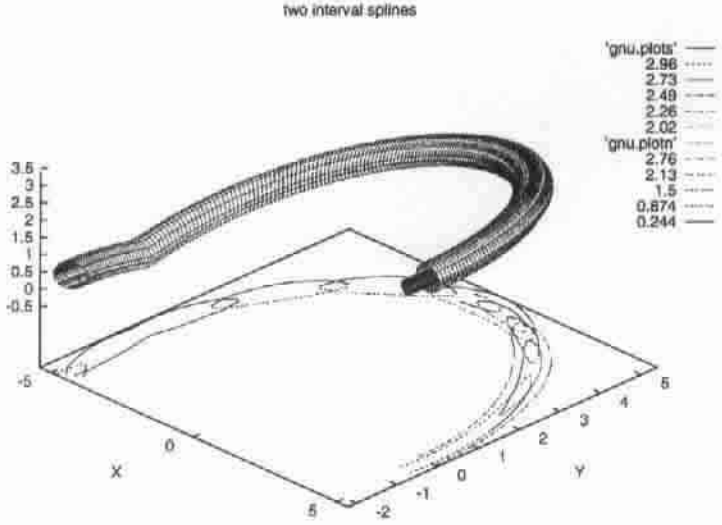

Figure 7. Opened tore with a vertical deformation.

inspection. Figure 9 includes range data from the scanner for the pocket in the cover plate. Figure 10 shows a CAD model for the pocket.

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 include the point from the scanner. Finally, we smooth the values from the radius 40 times and define the surface with

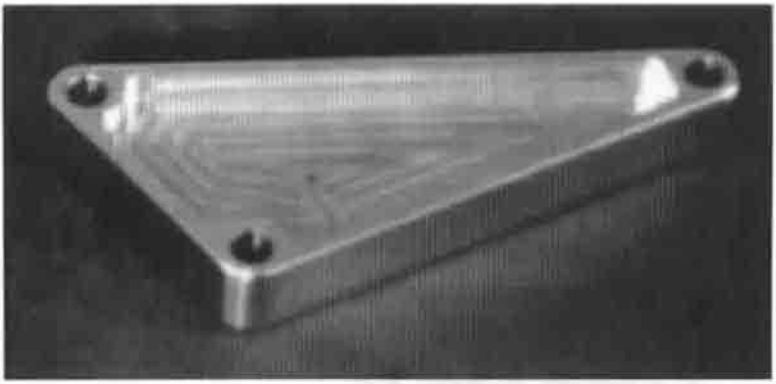

Figure 8. The machine part under inspection.

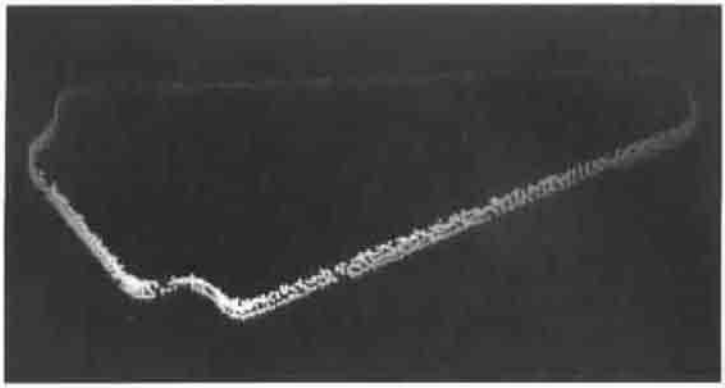

Figure 9. Range data for the pocket.

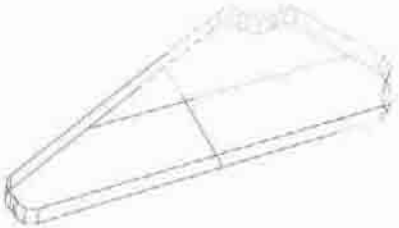

Figure 10. A CAD model for the pocket.

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.

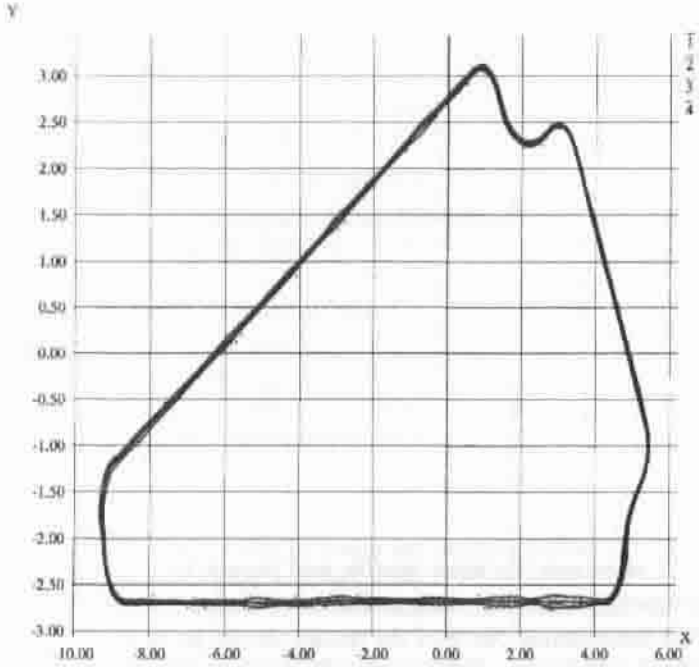

Figure 11. The points from the scanner and the computed offset surface: cutting of the inner pocket.

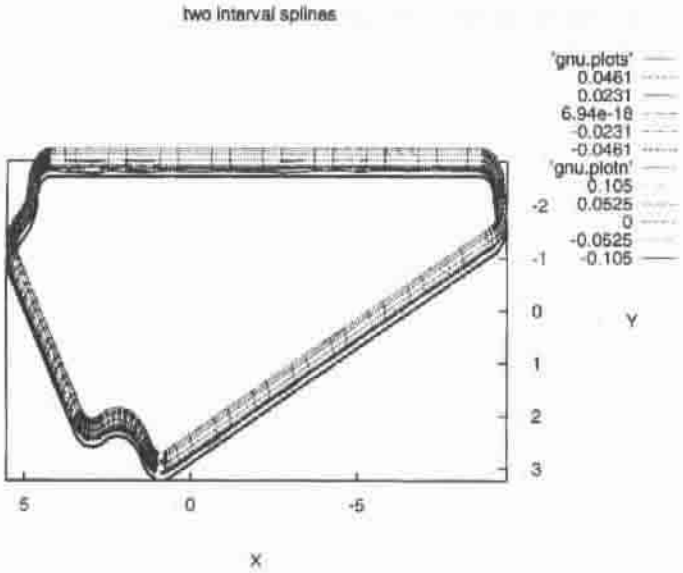

Figure 12. The 3D offset surfaces to compare.