VISUALIZATION OF TOLERANCE FOR MANUFACTURING

Tarek M. Sobh and Xiao Hong Zhu *

* *Department of Computer Science and Engineering*
University of Bridgeport
Bridgeport, CT 06601, U.S.A.
sobh@bridgeport.edu

Abstract: In this work we address the problem of visualizing the uncertainty in sensed data for manufacturing applications. Constructing geometric models for the objects from sense data is the intermediate step in a reverse engineering manufacturing system. Sensors are usually inaccurate, providing uncertain sense information. We construct geometric entities with uncertainty models for the objects under consideration from noisy measurements. This case study deals mainly with ways of visualizing the uncertainty in the geometric entities, in order to aid making later decisions on the geometry of the reconstructed parts.

Keywords: Sensor Systems, Robust Estimation, Uncertainty, Geometric Approaches

1. INTRODUCTION

Reverse engineering is a process that reconstruct a representation of a physical model, so that it can be reproduced identically. It is a new branch in the CAD/CAM field. Parts are manufactured according to blue prints, but when blue prints are not available, (such as, the part is too old, and its blue prints are missing), reverse engineering can be used to reproduce these parts. This can be achieved by the following two steps: sensing the part to construct its CAD representation and then manufacturing the part according to the representation. It is easy to see that the accuracy of measurement is the key to success in reproducing an accurate CAD model.

The accuracy of the measurement can be improved not only by improving the quality of measuring instrument, but also by optimizing sampling data. A reverse engineering system has been built and the measuring process is done by a vision sensor (B/W CCD camera).

In our reverse engineering system, we use a probabilistic approach to provide information for further measurements required to refine the CAD model, and also gives a quantitative measure of the accuracy of the current CAD model. The geometric reasoning on the probabilities of uncertain geometries can guide the sensor to perform focused measurements to allow for higher accuracy and efficiency. For instance, the slot (see figure 1) in mechanical engineering is a commonly used feature, and the parallelism of the two side planes is an important measurement.

The two side planes are based on sampling points from sensed data. Measurements of these points are not exact, therefore, these two planes that are constructed from these measurements, are planes with probabilities as the confidence measure. Consequently, the parallelism is no longer a definite relation, it has a probability distribution. If the confidence of the parallelism does not satisfy the manufacturing requirement, refinement of the two side planes is required, hence re-measuring of the points is performed.

Some work has been done in the probabilistic relationship between the geometric objects and their

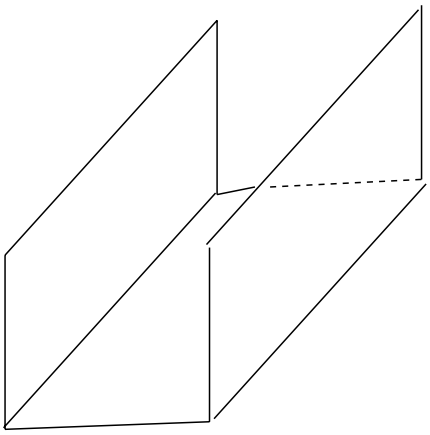

Fig. 1. Slot

relations, but the probability relations between the sampling points and geometric primitives have not yet been studied extensively. The geometric objects that this probabilistic geometric modeler is based on are constructed from sensing data. Therefore, study of the relation of the probabilistic characters of geometric objects and sensing data is very important. This work presents the study of these relations and ways of visualizing them. The work addresses the statistic geometric objects constructed from sensing data, relations of these statistic geometries, and the effect of decisions on its relative geometric objects.

2. RELATED WORK

Stochastic geometry has been systematically studied by mathematicians. In (**?**), mathematical theories of stochastic geometry are well studied, and uncertain geometric features can be represented as constrained functions. Classic examples of stochastic geometry can be found in (**?**). Kendall and Moran(**?**), describe a method of choosing distributions on geometric elements which provide a consistent interpretation of physical geometric elements.

Recently, research about sensing and uncertain geometry in robotics presents lots of ideas for handling uncertainty geometry. Hugh F. Durrant-Whyte in (**?**; **?**) has modeled the sensor in a manner that explicitly accounts for the inherent uncertainty encountered in robot operations. In Davidson's thesis(**?**), he made the important observation that arbitrary random geometric objects can be described by a point process in parameter space.

In computer-aided geometric modeling, methodologies for building a robust geometric modeler explores ways of handling the uncertain geometry caused by the imprecise computations. Arbitrary decisions are made, when uncertainty arises. In (**?**; **?**; **?**; **?**; **?**; **?**; **?**; **?**; **?**), three region tolerances are used to keep track of uncertainty caused by the computational error. In (**?**), arbitrary decisions are made and corresponding uncertainties are restricted.

3. REPRESENTATIONS FOR UNCERTAIN GEOMETRY

In geometric modeling, algorithms and representations for geometric objects are well developed, but the tolerance (uncertainty of geometry) has not yet been well defined. In (**?**), a geometric object is represented by boundary and hybrid representations, associated with a tolerance representing the uncertainty of the geometry.

Based on the representations that has been developed and used in (**?**), a representation for uncertain geometry is developed as follows:

An uncertain geometric object is represented in two parts: a geometric description, and a probabilistic distribution of geometry. The geometric description is a parameter vector, and the probabilistic distribution of geometry is a vector of the same dimensions as the geometric description, but with corresponding probabilistic distributions of the parameters.

For instance, a plane can be specified as a equation: $(A, B, C), (f_a, f_b, f_c)$, where (A, B, C) is the geometric description and $z = Ax + By + C$. (f_a, f_b, f_c) is the probabilistic distribution of geometry, and also can be specified in another form: $(P, V), (f_p, f_v)$, where P is a base point, and V is the normal vector of the plane. f_p is the uncertainty of the base point, and f_v is the uncertainty of the normal vector. It can be proved that f_p and f_v can be computed from f_a, f_b, f_c, and P, V can be computed from (A, B, C). By defining f_a, f_b, f_c, different types of probability distributions can be handled by this representation.

4. EXPERIMENT ON STATISTIC GEOMETRIC OBJECTS CONSTRUCTED FROM SENSING DATA

The geometric objects being dealt with are constructed from the sensed data. How the distribution of sensing data affects the uncertainty of the geometry is the basis for defining the distributions of the geometry. In this section, the uncertainty of the plane relating to the sensing 3D coordinates is studied. A set of discrete sensing data is used to perform the computations.

4.1 *Best Least Square Fit*

In order to reduce the random error, usually, n sampling points are measured to defining a

plane, yet the points have certain probability distributions which mainly depend on the measuring machine, the n points are independent random events. Therefore, a best least square fit method for computing the plane parameter is used. This approach gives the maximum liklehood result, and confidence on the sampling data to be a plane.

Assuming that input data is (x_i, y_i, z_i), where x_i, y_i, z_i are either fixed values, or probability functions. They can be either independent, or correlated. Explicit function definition for a plane in 3D will be $z = Ax+By+C$. If there are n points, the best least fit plane should be the solution of the following equation set.

$$Z = P \bullet X$$

Because P is an n x 3 matrix, and X is a 3 x 1 matrix, $rank(P) = 3$, and $n \geq 3$, solution of X is unique. When $n > 3$, the solution X is a best least square fit.

$$X = (P^T \bullet P)^{-1} \bullet P \bullet Z$$

Or in the other form:

$$\begin{aligned} A &= f(x, y, z) \\ B &= g(x, y, z) \\ C &= h(x, y, z) \end{aligned}$$

Where $x \in [x_1, x_2]$, $y \in [y_1, y_2]$, $z \in [z_1, z_2]$ are discrete. Function f, g, h are non-linear functions. To compute the probability distribution of A, B, and C, exhaustively computing values of f, g, and h, will provide the discrete probability distribution array for A, B, and C.

From the above mathematics, we can see that the computation complexity is exponential. If m is the number of distribution values and n is the number of sampling points, this above computation will be performed $(3^m)^n$ times.

4.2 *Sensing Data and its Corresponding Results*

The sensing data is modeled by discrete points with their corresponding probabilities. Normally, a point in 3D is represented as (x, y, z), but for this sensing data, x, y, and z, are no long a single value, they are distributions as shown in figure 2.

Due to the computational complexity, and the generality of the problem, a three distribution values data set is used for experiments. The resulting planes (A, B, C) along with their distributions are computed. Graphs of A,B, and C distributions are approximated by the following computations.

What we want to get is the concept of the $f(x)$ shape. The data we computed are discrete state vector (A, B, C) and its probability. $P(x_i < x < x_{i+1}) = \int_{x_i}^{x_{i+1}} f(x)$ is computed and plotted, where x can be A, B, or C. and $x_{min} \leq x_i \leq x_{max}$. In order to smooth the curve, an overlapped set of x_i is used. In the result figures, the x axis are the values of A, B, C respectively, and the y axis are the corresponding probability of that value.

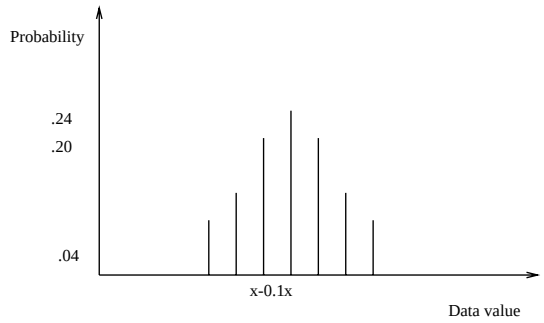

Fig. 2. Sensing Data

Test 1: Uniform distribution: the sensing data is shown in figure 3.

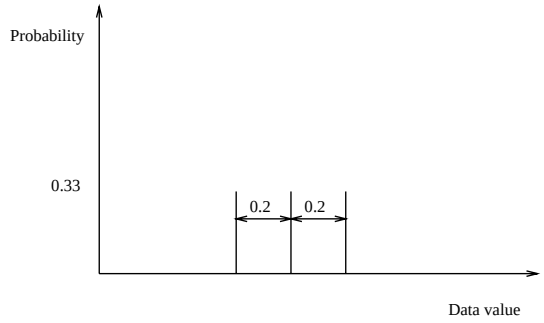

Fig. 3. Uniform Distribution

There are a total three points with such distributions, planes defined by these points are computed.

Test 2: Gaussian distribution: the sensing data is shown in figure 4

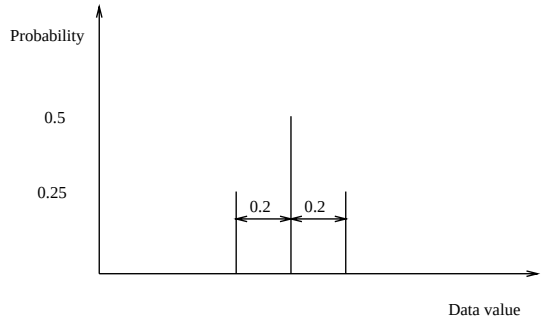

Fig. 4. Gaussian Distribution

There are a total of three points with such distributions, the planes defined by these points are computed.

Next, we discuss an algorithm for visualizing the groups of resulting probabilistic planes from the sensed data.

5. VISUALIZING SENSED DATA FOR PLANAR SURFACES

5.1 *Introduction*

Problem : Given sensed data for planar surfaces the goal is to design an algorithm for grouping this data and then finding effective ways of visualizing these groups.
Each actual planar surface is represented by a very large number of planes of the form :
Z = a_i X + b_i Y + c_i
The sensed data consists of :

- a_i,b_i,c_i values for each of the above planes.
- Probability of each plane represented by (a_i,b_i,c_i) of being the actual planar surface.

Given the above sensed data we want to find a way to cluster these (a_i,b_i,c_i)'s according to their spatial proximity in (a,b,c) 3D space and then to find a suitable way of visualizing these clusters.

5.2 *The Algorithm*

(1) Use a **Recursive Algorithm** to find all pairs of points (α_i,α_j) such that the distance between them does not exceed a small number ϵ.
(2) The above step returns a set of clusters each having two elements such that the distance between these two elements does not exceed a small number ϵ. In this step we group all those clusters having one common element i.e. if (0,1),(0,2),(0,3) are some of the clusters returned by the previous then this step would combine these three clusters to form a final cluster (0,1,2,3). So this step in effect reduces the number of clusters returned by the previous step by combining all those clusters which have points at a distance not more than ϵ from a common point which is the **SEED** for that cluster.
(3) The goal of this step is to combine clusters returned by the previous step by identifying the **Seed** for each cluster. In this step we represent each cluster as a n band of planes where n is the number of (a,b,c) values in this cluster. The planes are represented by triangles. Each cluster is represented in a different color. The thickness of a set of triangles representing a cluster are a measure of the probability of this set of a,b,c values of being that of the actual plane.

5.2.1. ***Detailed Description of the Algorithm*** **Step 1:** Our Algorithm uses the **n-dimensional BILS Method** (**?**).
n-dimensional BILS Method
Suppose we wish to solve an n dimensional problem : for a set S = α_1,α_2,...,α_m of m points in n dimensional space report all coincident pairs (α_i,α_j) such that α_i and α_j (i < j) are points in S and the distance between α_i and α_j does not exceed a given small number ϵ.
Let α_i in S be represented as an ordered set of n floating point numbers (β_{i1},β_{i2},...,β_{in}). So, α_i = (β_{i1},β_{i2},...,β_{in}), for i = 1,2,... m;
In each axis direction k, we find the lower bound lb_k and the upper bound ub_k. Thus we can find the bounding box which encloses S by n intervals [lb_j,ub_j], where j = 1,2,...,n.
The algorithm computes the dividing axis k as R modulo n where R is the current recursion depth and n is the dimension of the space. The kth interval of the bounding box B is subdivided into two subintervals of equal size [lb_k,midpoint] and [midpoint,ub_k]. Hence the bounding box B is subdivided into two boxes B_1 and B_2.
The original List of indices L is subdivided into two sublists L_1 and L_2 so that L_1 contains the indices of all points which are in B_1 and L_2 contains the list of all indices which are in B_2. The List is subdivided into two lists L_1 and L_2 by simply comparing the kth co-ordinate β_{ik} of α_i with two threshold values (midpoint - ϵ) and (midpoint + ϵ) for each element i in the List. The algorithm allows the index list subdivision method to be "binary" subdivision regardless of the dimension of the space.
The above algorithm as given by (**?**) does not give a way on deciding on ϵ and the value of ϵ is constant regardless of the data points. Our algorithm is adaptive because the value of ϵ changes for different clusters depending on the data points. In our algorithm we set ϵ to a small value initially and keep increasing ϵ and calling the above recursive algorithm till all the a,b,c values are in some cluster. At the end of this step each cluster has a pair of points at a distance not more than ϵ from each other where ϵ may be different for different clusters.

Our Algorithm :

ϵ = 1;
DONE = FALSE;
While NOT DONE
DONE = TRUE;
rec_depth = 0;
Initialize the *bounds* array to the minimum and maximum values along the X,Y and Z directions.
bils(index,num_elements,bounds,rec_depth);
For each element i in the *index* array
If any element is not in a cluster
DONE = FALSE;
increment ϵ by 1;

end of while

In the above algorithm :

- **The** *bounds* **array has the minimum and maximum value in all axis directions. In our case we are working with 3D points and so X,Y and Z are the axis directions.**
- **The** *index* **array has the list of all indices.**
- *num_elements* **is the total number of (a,b,c) values.**
- *rec_depth* **is the recursion depth.**

The above algorithm calls the following *bils* **algorithm :**
bils(index_list,num_index,b,curr_rec_depth)
If($(num\_index <= \epsilon)$ **OR** $(curr\_rec\_depth == MAX\_DEPTH)$**)**
search(index_list,num_index);
else k = curr_rec_depth modulo n;
n1 = 0; n2 = 0;
Let the mean of lb_k **and** ub_k **be** *midpoint*, **where** lb_k **and** ub_k **are the lower and upper bounds of the** *kth* **dimension.**
For each element i **in the** *index_list* **Let k[i] be the** *kth* **co-ordinate of a point** α_i **in S.**
If k[i] <= (*midpoint* **+** ϵ**)**
add i **to** *list1*; **increment** *n1* **by 1.**
If k[i] >= (*midpoint* **-** ϵ**)**
add i **to** *list2*; **increment** *n2* **by 1.**
If($n1 > 1$**)**
Set the *kth* **interval of** b **to be** $[lb_k, midpoint]$;
bils(list1,n1,b,curr_rec_depth + 1);
If($n2 > 1$**)**
Set the *kth* **interval of** b **to be** $[midpoint, ub_k]$;
bils(list2,n2,b,curr_rec_depth + 1);

In the above algorithm :

- *index_list* **array has the list of all indices.**
- *num_index* **is the total number of (a,b,c) values.**
- b **has the minimum and maximum values in each axis direction.**
- *curr_rec_depth* **is the current recursion depth.**
- S **is the list of (a,b,c) values.**

Step 2: This step combines clusters returned by the previous step and the criteria for combining is that the elements in a final cluster should be within a distance not more than ϵ from the *seed* of the cluster.

i = 0; j = 1; final_cluster_count = 0;
Let the first and second element in cluster i **be** $cluster_{i1}$ **and** $cluster_{i2}$ **respectively.**
Let the first and second element in cluster j **be** $cluster_{j1}$ **and** $cluster_{j2}$ **respectively.**
For each cluster i **in the cluster array**
If $cluster_{i1}$ **is not in any** *final_cluster*
SET SEED = $cluster_{i1}$;
Put $cluster_{i1}$ **in** *final_cluster k.*
If $cluster_{i2}$ **is not in any** *final_cluster*
Put $cluster_{i2}$ **in** *final_cluster k.*
else If $cluster_{i2}$ **is not in any** *final_cluster*
SET SEED = $cluster_{i2}$;
Put $cluster_{i2}$ **in** *final_cluster k.*
For each cluster j **in the cluster array (i <** j**)**
If($cluster_{j1}$ **== SEED)**
If $cluster_{j2}$ **is not in any it final_cluster**
Put $cluster_{j2}$ **in** *final_cluster k.*
increment final_cluster_count;
increment i; **SET** $j = i + 1$;

Step 3: Once we have the final clusters i.e. for each cluster we have a set of a,b,c values. These a,b,c values represent planes of the form
Z = a X + b Y + c
For each a,b,c value in a cluster we find the intersection of the plane it represents with the three principal axes X,Y,Z.
Intersection with Z axis put X = 0, Y = 0 and so we get (0,0,c).
Intersection with X axis put Z = 0, Y = 0 and so we get (-c/a,0,0).
Intersection with Y axis put X = 0, Z = 0 and so we get (0,-c/b,0).
So we now have three points on the plane i.e. the intersection with X,Y and Z axes. We therefore represent each plane as a triangle. A cluster is represented as a band of triangles. Each plane has a probability of being the actual plane. Now a cluster has a number of such planes. A cluster with more planes will have a higher cluster probability and this is indicated by the thickness of the triangles.

6. RELATIONS OF STATISTIC GEOMETRIES AND ITS EFFECT ON RELATIVE GEOMETRIES

As mentioned in the introduction, the goal of this probabilistic approach is to feedback control to the sensing devices to measure the physical model and give a quantitative confidence measurement for the CAD model. Some relations of these uncertain geometries are computed, and results are computed with their uncertainty distributions.

Basically, geometric relations are set relations. For example; intersecting, coincidence, incidence, apartness, and parallelism. Because of the uncertainty of the geometries, these relations are not definite, they are decisions with certain confidence, also, this confidence can be specified by

its probability. For instance, a point incident on a plane, can be computed as a point incident on the plane with 0.9 probability. This provides reasoning based on probabilities.

A feedback computation of a plane that is supposed to be collinear with a given plane is studied. A program that takes the output discrete planes along with their probabilities is implemented, and the cases of parallel and collinear statements are computed with their probabilities. Some examples of parallelism and collinearity have been tested. For example, collinearity and parallelism of the uniform distribution planes (as described above has been tested). The probability for parallelism is 0.824719, for collinearity is 0.334722. The parallelism and collinearity of the planes of the three points Gaussian distribution and the uniform distributions have also been tested. The parallelism is 0.66730846, and the collinearity is 0.27099140. (the tolerance for testing them is the square distance less than $10e^{-2}$).

A	B	C	P(probability)
1.034723	-0.961805	2.386458	0.33
1.036584	-0.966461	2.391768	0.34
1.038042	-0.970109	2.395926	0.33

If we assume that the plane constructed from the uniform distribution sensing data is decided to be collinear to the plane defined by the above table, then, its distribution is recomputed as follows: among this plane set, the plane instances which are not collinear with any of the plane instances in the given plane set, is discarded.

7. CONCLUSIONS

Based on real sensing data, the probability of the geometry of the objects under consideration is computed and visualized. This provides us with the capability to define the probability distribution of the geometry based on robust computations as opposed to noisy measuring instruments. The relations between uncertain geometries are dependent on the uncertainty of geometries. Quantitative measurement for the constructed CAD model can thus be computed and decisions can be made with the help of the visualization modules.

8. REFERENCES

Bruderlin, Beat (1990). Detecting ambiguities: An optimistic approach to robustness problems in computational geometry. Tech. Rep. UUCS 90-003 (submitted). Computer Science Department, University of Utah.

Bruderlin, Beat (1991). Robust regularized set operations on polyhedra. In: *Proc. of Hawaii International Conference on System Science.*

Bruderlin, Beat and Shiaofen Fang (1992). Intuitionnistic geometry: A new approach for handling geometric robustness. *submitted to: International Journal of Computational Geometry and Applications.*

Durrant-Whyte, Hugh F. (1986). Concerning uncertain geometry in robotics. *IEEE J. Robotics and Automation.*

Durrant-Whyte, Hugh F. (1988). *Integration, Coordination and Control of Multi-Sensor Robot Systems.* Kluwer Academic Publisher.

Fang, S., B. Bruderlin and X. Zhu (1993). Robustness in solid modeling – a tolerance-based, intuitionistic approach. *To appear: Computer-Aided Design (Special Issue on Uncertainties in Geometric Computations.*

Fang, Shiaofen (1992). Robustness In Geometric Modeling. PhD thesis. University of Utah.

Fang, Shiaofen and Beat Bruderlin (1991). Robustness in geometric modeling – tolerance based methods. In: *Computational Geometry – Methods, Algorithms and Applications, International Workshop on Computational Geometry CG'91.* Springer Lecture Notes in Computer Science 553, Bern, Switzerland.

Fang, Shiaofen and Beat Bruderlin (1992). Robust geometric modeling with implicit surfaces. In: *Proc. of International Conference on Manufacturing Automation, Hong Kong.*

Fang, Shiaofen, Xiaohong Zhu and Beat Bruderlin (1992). Robustness in solid modeling - a tolerance based, intuitionistic approach. Tech. Rep. UUCS 92-030 (submitted). Computer Science Department, University of Utah.

J.Bertrand (1907). *Calcul des Probabilites.* Paris.

Kendall, M.G. and P.A.P. Moran (1963). *Geometrical Probability.* Griffin.

Leon, Midori K. De (1993). Boolean Operations on Polygon Meshes. PhD thesis. Texas A&M University.

R.Davidson (1968). Some Arithmetic and Geometry in Probability Theory. PhD thesis. Cambridge University.

Zhu, Xiaohong (1993). Consistent geometric modeling approaches. Master's thesis. University of Utah.

Zhu, Xiaohong, Shiaofen Fang and Beat Bruderlin (1993). Obtaining robust boolean set operation for manifold solids by avoiding and eliminating redundancy. In: *Proc. of Second Symposium on Solid Modeling and Applications.*