Design Rules for Tolerance-Insensitive and Multi-purpose Fixtures*

Yan Zhuang
Computer Science Department
University of California
Berkeley, CA 94720
yzhuang@cs.berkeley.edu

Ken Goldberg
IEOR Department
University of California
Berkeley, CA 94720
goldberg@ieor.berkeley.edu

January 1997

Abstract

The manufacturing and installation of fixtures in a factory can be costly, especially for high-volume production runs. Once fixtures are installed, part shape can vary due to: (1) tolerance variations because of inherent imprecision in manufacturing processes, and (2) redesign. Variations of type (1) occur constantly and are usually small; variations of type (2) can be large but occur at few distinct points in the product life cycle. For both cases, it is often desirable to *reuse* one fixture for all variations of part shape.

In this paper we address both types of variation in the context of modular fixtures. For tolerance variation (1) , we give a method for comparing fixtures in terms of how much geometric tolerance they will permit before failing. For design variation (2), we identify *design rules* that can rapidly check if a proposed part design is consistent with a given previous fixture. The idea behind the design rule is that it can be incorporated into the CAD design cycle to rapidly evaluate manufacturability in a manner analogous to the geometric design rules so successful in VLSI design. We also present an efficient algorithm to find multi-purpose fixtures that can fixture multiple distinct parts, using the geometric hashing technique [10, 11].

Key words: Fixturing, design rules, tolerance, geometric hashing.

1 Introduction

In today's industry, many products' lifetime is short due to high competition. Redesign of an existing product is quite often. To reduce the cost and time to bring a new design to production, manufacturers have to be able to quickly reconfigure the assembly line for the redesigned or new product. Fixturing as an integrated component

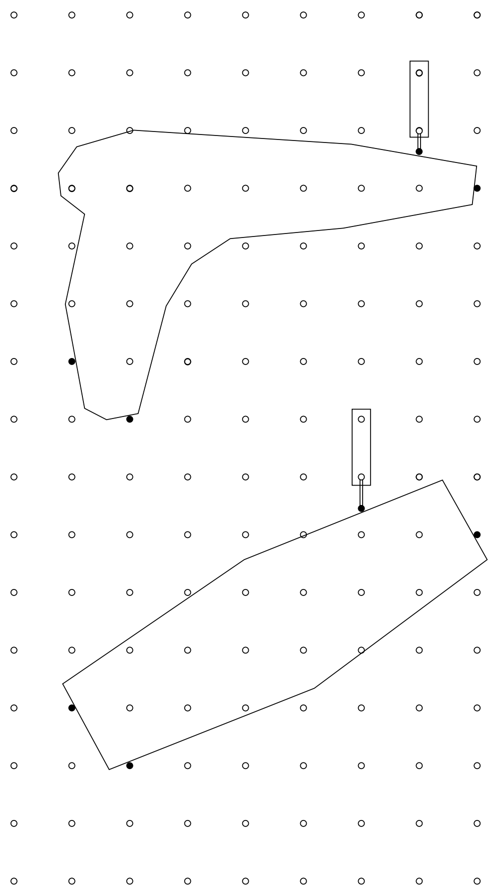

Figure 1: Two different parts share the same set of 3 locators.

*This work was supported in part by NSF Awards IRI-9612491, Ford and Presidential Faculty Fellow Award IRI-9553197 to Prof. Goldberg.

of the manufacturing process usually also has to be redesigned or reconfigured for the new product.

A fixture is a device that immobilizes the part for machining, assembly, inspection, etc. A modular fixture [6] system offers a regular lattice of holes, on which locators and clamps can be precisely mounted.

Manufacturing processes are inherently imprecise, therefore the part shape can vary. A computed fixture or assembly plan can fail due to the inherent error in part's shape [12, 8]. This kind of failure could be very costly, therefore it should be detected as early as possible.

To reduce the cost of manufacturing and installation of fixtures, it is desirable to *reuse* fixtures. In this paper, we address 3 types of *re-usability* of fixtures: (1) Given multiple parts, and the corresponding set of fixtures, we would like to find the common set of fixtures that can be applied to all the given parts. We will refer to this common set of fixtures as the *multi-purpose fixtures* (Figure 1). (2) Given a part and its fixture, we would like to redesign the part while still being able to reuse the same fixture to the redesigned part. The question is what redesign rules we have to follow. (3) Given a part with tolerance variations, we would like to find the condition to reuse the same fixture under geometric uncertainty.

Design rules are essential to achieve high yields in VLSI package assembly [24]. The idea is that design rules must be generated for the particular package type used and must be compatible with the assembly equipment to be used, and that these decisions should be made as early as possible.

In this paper, we integrate the fixturing into the product design process. We identify *design rules* that can rapidly check if a proposed part design is consistent with a given previous fixture. The design rule can be incorporated into the CAD design cycle to rapidly evaluate manufacturability in a manner analogous to the geometric design rule so successful in VLSI design [24]. The design rule also gives a necessary condition for a computed fixture to work under geometric tolerance [22].

For a given machining or assembly operation, one face of the part is generally in contact with the supporting plane. We need to immobilize the part on the supporting plane. In this paper, we refer to a part as its 2D projection onto the supporting plane. In particular, we consider polygonal parts. We assume that a planar modular fixture configuration consists of 3 locators and 1 clamp, which provides form closure. We also assume that all contacts are frictionless contacts and the contact point is within the interior of the edge. Note that the frictionless assumption is conservative, because a frictionless fixture is always a valid fixture with friction.

In section 2, we briefly summarize the previous study in the related areas. In section 3, we present a linear algorithm for computing the multi-purpose fixtures, using the geometric hashing technique. In section 4, we derive the design rule to ensure the 3 locator contact after modifying the part. In section 5, we briefly discuss the application of the design rule to a limited case of fixturing under geometric uncertainty. In section 7, we present some potential extension of the work.

2 Related Work

Fixturing is closely related to grasping. The purpose in both cases are to immobilize the part. Modular fixturing is however different from grasping because the locators are restricted to the discretized holes on the regular lattice.

The form closure condition defined by [23] gives the sufficient condition for immobilization. A set of contacts is said to provide form closure if infinitesimal motions are prohibited. Equivalently, a set of contacts provides form closure if the corresponding set of wrench vectors positively span the wrench space. In planar case, the wrench space is $\mathcal{R}^3$. The first two components of the wrench vector are the X and Y forces in the plane and the third component is the moment about an arbitrary point on the plane. Since we assume frictionless contacts, the force vector (the first 2 components of the wrench vector) has to be normal to the boundary of the part [17].

Goldman and Tucker [5] show that at least $n+1$ vectors are necessary to positively span $\mathcal{R}^n$, and there exists a set of $n+1$ vectors to positively span $\mathcal{R}^n$. For grasping and fixturing, Markenscoff *et. al*[13] show that there exists a set of 4 wrenches to provide form closure for any piecewise-smooth planar part, except for surface of revolution.

Asada and By [1] show how to determine whether a given fixture design provides total constraint of a rigid body, as well as loading accessibility before clamping. Mishra *et. al*[14] is the first one that applies work on grasping to fixturing using an regular lattice. Nguyen [15] gives a geometric test for form-closure, given 4 coplanar contacts. Pertin-Troccaz [20] gives a good survey of grasping literature.

Recently there is a surge of research on fixturing. Brost and Goldberg [2] present the first complete synthesizing algorithm that guarantees to find a fixture, consisting of 3 locators and 1 clamp, for any given polygons, if one exists, and return failure otherwise. Wallack and Canny [25] present a complete algorithm for a fixturing model using 4 locators on a split lattice that can open and close like a vice. Penev and Requicha [19] present their study on fixture foolproofing: given a fixture consisting 3 locators and 1 clamp, where should the blocking pins be inserted to insure that the part can be loaded in only one desired pose? Overmars *et. al*[18] propose a new class of planar fixtures that includes flat edge-contact and present a complete algorithm to find such fixture. Wentink *et. al*'s most recent work on fixture planning can be found in [26]. Brost and Peters [3] present an algorithm that automatically designs fixtures and assembly pallets to hold three-dimensional parts.

Lamdan and Wolfson [10, 11] introduce the idea of geometric hashing. It uses an indexing approach based on transformation invariant representation to quickly find matched surface patches. Geometric hashing is first applied to the object recognition problem in computer vision [27]. Fischer *et. al*[4] and Norel *et. al*[16] apply the geometric hashing technique to the docking problem in molecular biology: detect the optimally matched ligand and receptor surfaces. In this paper, we will apply geometric hashing to find multi-purpose fixtures.

Geometric tolerancing is concerned with the specification of geometric shapes for use in the manufacture of mechanical parts. Requicha [21, 22] gives mathematical definition of tolerance specifications, such as the zone tolerance that we are going to use in section 5. Yap [28] discusses the classification problem in the metrology of geometric tolerancing. Latombe and Wilson [12] present an efficient algorithm that decides whether a product made of toleranced parts admits a guaranteed assembly sequence. Joskowicz *et. al*[8] present a general method for worst-case limit kinematic tolerance analysis: computing the range of variation in the kinematic function of mechanism from its part tolerance specifications.

3 Multi-purpose Fixtures

For a planar workpiece, we often have multiple ways to fixture it. Indeed Brost and Goldberg [2] present a complete algorithm that computes the complete set of fixtures for any given polygonal planar part. Redesign and reconfiguration of fixtures does not only cost money, but also time. Therefore it would be desirable to re-use the same fixture for multiple parts. We call such a fixture a *multi-purpose fixture*.

In this section, we consider particularly the modular fixture that consists of three locators and one clamp *(3L/1C)*, all mounted onto a regular lattice of holes. Given multiple sets of such fixtures,[1] we present an efficient algorithm to find the largest set of multi-purpose fixtures.

To re-use a fixture, we have to be able to re-use the three locators and then we have to be able to find a clamp location to achieve form closure for each part.[2] We denote the locator arrangement of each fixture by a triplet (l_1, l_2, l_3). For each set of 3L/1C fixtures, we denote the corresponding set of locator triplets by T_i. The largest set of locator triplets that can be re-used for k parts are denoted by $T = \sqcap_{i=1}^{k} T_i$. The algorithm that finds the largest set of multi-purpose fixtures consists of the following two steps:

1. Given k parts and the corresponding k sets of 3L/1C fixtures, find $T = \sqcap_{i=1}^{k} T_i$, where $T_i = \{(l_1^j, l_2^j, l_3^j) | j = n_1, ..., n_i\}$ is the corresponding set of locator triplets.

2. For each triplet $t \in T$ and for each part, we compute the clamp location that achieves the form closure. For each triplet t, if there exists a part that no clamp location can be found for form closure, then t is deleted from T. At the end, T contains the largest set of multi-purpose fixtures.[3]

In section 3.1, we present an efficient algorithm to compute the intersection mentioned in step 1. There are many ways to compute clamp location and verify the form closure, which will be briefly discussed in section 6.

3.1 Geometric Hashing

First let us consider 2 parts, whose sets of locator triplets are T_1 and T_2, respectively. Later we will extend it to the general case of multiple parts.

Note that each locator triplet uniquely determines a triangle. A triplet $t \in T_1$ is re-usable if and only if there exists a triplet $s \in T_2$, such that s can be obtained by applying compositions of translations and rotations to t. We refer to such triplets as *equivalent* triplets. Note that we exclude reflection in the transformation because it is often the case that the part has to face up (or face down but not both) during operations, such as machining and assembly. Namely we cannot assume that we are allowed to flip the planar part. Our goal is to compute the largest set of equivalent triplets of T_1 and T_2: $T_1 \sqcap T_2$. Note that it is possible that T_1 contains equivalent triplets in itself. However by our definition, at most one of those equivalent triplets should appear in $T_1 \sqcap T_2$.

A naive solution is to compare each element in T_1 with each element in T_2. This will require an $O(nm)$ running time, where $|T_1| = n$ and $|T_2| = m$. The experimental data of [2] shows that the value of n and m can be in the range of thousands. Therefore we would like to have a more efficient algorithm to find $T_1 \sqcap T_2$.

The more efficient algorithm is to use an indexing approach based on transformation invariant representation. The algorithm efficiently scans T_2 and detects

[1] For example, the set of fixtures of each polygonal planar part can be the output of the Borst-Goldberg algorithm [2].

[2] The clamp location could be different for each part, even if all parts share the same locator triplet.

[3] Now each triplet in T is also linked with a list of clamp locations. We abuse the notation a little here, to avoid introducing new notations for the final set of multi-purpose fixtures.

equivalent pairs of triplets in T_2 and T_1. This idea is borrowed from the object recognition problems in computer vision. It is usually referred to as geometric hashing [10, 11]. Now the question is how to construct the index of each triplet such that two triplets have the same index into the hash table if and only if they are equivalent.

Each triangle is uniquely determined by the lengths of its three edges. We denote the three lengths as d_1, d_2 and d_3. Without loss of generality, we assume that $d_1 = \max\{d_1, d_2, d_3\}$. Furthermore we assume, without loss of generality, that d_1, d_2 and d_3 follow the counter-clockwise order. Since we assume that locators are mounted onto a regular lattice, the square of d_i, $i = 1, 2, 3$, is an integer in units of grid spacing. Therefore we can construct a long integer by storing d_1^2 into the higher bits, d_2^2 into the middle bits, and d_3^2 into lower bits, as shown in figure 2. We denote this long integer by $k(d_1, d_2, d_3)$, and we refer to it as the "key" of the corresponding triplet.

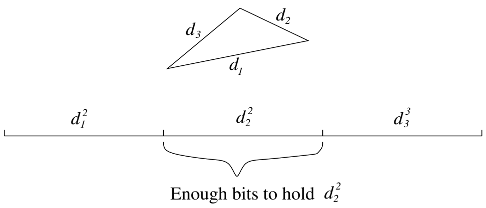

Figure 2: A unique integer (*key*) can be constructed for each triangle.

Each class of equivalent triplets has a unique key. The ordering of d_i's ensures that the reflection of a triplet does not have the same key. It defines a bijective mapping between integers and the family of sets of equivalent triplets.

Direct addressing using the keys constructed above is impossible for most of today's computers. Just to illustrate the memory requirement for direct addressing, let us consider a regular grid lattice of size r by r. This means that the square of maximum length of each side of the triangle is $O(r^2)$. This requires a direct-address table as large as $O(r^6)$. Since the size of the given set of fixtures is relatively much smaller than that of the direct-address table, we can build a hash-table. An triplet with key k is stored in entry $h(k)$, where h is the hash function. Each entry of the hash table holds the information of the triplet and one bit that indicates whether this entry has been hit before. The computation of $T_1 \sqcap T_2$ consists of two key steps:

1. **Construction**: For each element of T_1, compute its corresponding key, $k(d_1, d_2, d_3)$, as described above. Store this triplet into the corresponding entry, using the hash function, and set the bit to 1 (initially 0).

2. **Query**: For each element of T_2, compute its key and access the corresponding entry of the hash table. If the bit in the entry is 1, output the triplet in the entry, otherwise do nothing.

The choice of the hash function is not unique. A good hash function should map keys evenly to hash-table entries. For example, one possible hash function is as following:

$$h(k) = (int)(10^4 d_1 + 10^2 d_2 + d_3) \tag{1}$$

where $k = k(d_1, d_2, d_3)$.

Alias problem may occur. Namely it is possible that two *non-equivalent* triplets with different keys are hashed into the same entry of the hash table. Theoretically in the worst case, all triplets in T_1 can be hashed into the same entry, which means we have to compare elements in two sets one by one. However practically, they are usually sparsely distributed in the hash table because the size of the hash table is much bigger than that of T_1. When alias does occur, we can resolve the collision by the classical technique such as chaining. Excellent analysis of hashing algorithms can be found in [9].

3.2 Complexity

The construction of the hash table takes $O(n)$ expected time and the query takes $O(m)$ expected time. Overall it takes $O(n + m)$ expected time to compute $T_1 \sqcap T_2$. Therefore we improve the time complexity from quadratic to linear. In the worst case, we have $|T_1 \sqcap T_2| = \min(|T_1|, |T_2|)$, therefore this geometric hashing algorithm's time bound is both tight and optimal.

3.3 Extension to Multiple Sets

Note that the geometric hashing algorithm presented in section 3.1 can be easily extended to arbitrary number of sets T_i, $i = 1, ..., k$. The only changes in the algorithms are listed as following:

1. Instead of having a bit to indicate the hit in each hash-table entry, we use a *counter* to keep track the number of hits so far.

2. Construction step remains the same.

3. **Query**: For each set T_i, $i = 1, ..., k-1$, for each of its elements, compute its key and increase the counter in the corresponding entry by 1. For each element of T_k, compute its key and access the hash table, output the triplet in the corresponding entry if the counter has the value $k - 1$.

The expected running time will be $O(\Sigma_{i=1}^{k} n_i)$, where $n_i = |T_i|$, which is linear in size of input.

4 Design Rules for Tolerance Insensitive Fixtures

In manufacturing industry, we often have to redesign a product. Usually a new design requires a completely different fixture design, which costs both money and time. This tempts us to consider the following problem: *given a planar polygonal part, and its fixture which consists of 3 locators and 1 clamp, is it possible to specify how much the part shape can be modified such that the redesigned part can still use the same set of locators?*[4]

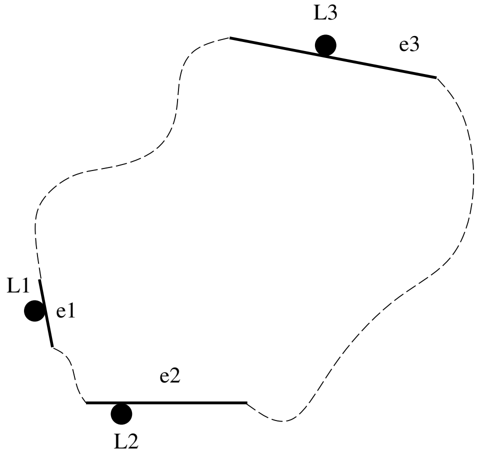

Figure 3: How much can we modify $e3$ such that we can still achieve 3 point contact?

We assume that the designer will redesign the part by modifying one edge at a time. Apparently the designer can modify the non-contacting edges at his or her will, without affecting the fixture. The real question is: *given 3 locators and their corresponding contacting edges, if we keep 2 of the edges, how much we can modify the third edge while still being able to achieve 3 point contact? (Figure 3)* Note that the dashed curves in figure 3 stand for all the edges connecting the three edges in consideration. Without loss of generality, we expand the part by the radius of locators using a Minkowski sum operation. This allows us to consider each locator as an ideal point. The original part contacts a locator if and only if the corresponding edge of the expanded part passes through the corresponding point. In the rest of this section, we treat expanded part as the part.

4.1 The Contact Locus

Given two locators and their corresponding edges, the part can move while maintaining the two point contact. During the motion, the third edge in consideration sweeps out a region within which it can achieve the third point contact. This motion is difficult to characterize and the region swept out by the third edge has complicated shape. To simplify the analysis, we can instead consider the motion of the triangle, formed by the three point locators, in the moving frame attached to the moving part. The triangle's two vertices can slide on two edges of the part, while the third vertex sweeps out a curve in the moving frame. If this curve in the moving frame intersects the part's boundary, there exists a 2D rigid body motion to achieve 3 point contact. Note that all edges are fixed in the moving frame. Now the question becomes what that curve is in the moving frame?

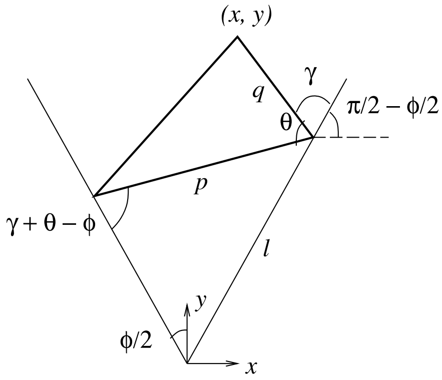

Figure 4: It is an *elliptic* curve.

We follow [7] to characterize the curve swept out by the third locator. Without loss of generality, we can assume that the triangle is sliding on a cone, which has its apex at the origin and is symmetric about y-axis (Figure 4). Furthermore, given the cone and the triangle, the following values are constants: ϕ, p, θ and q. The position (x, y) of the third vertex of the triangle is uniquely determined by a single parameter γ. By the Sine Law we have

$$\frac{l}{\sin(\theta+\gamma-\phi)} = \frac{p}{\sin\phi} \tag{2}$$

Then we can have the following expressions for (x, y) given γ.

$$\begin{cases} x = l\sin\frac{\phi}{2} + q\sin(\frac{\phi}{2}-\gamma) \\ y = l\cos\frac{\phi}{2} + q\cos(\frac{\phi}{2}-\gamma) \end{cases} \tag{3}$$

This is a linear system of $\cos\gamma$ and $\sin\gamma$. By some manipulations we can simplify it to the following:

$$\begin{cases} \sin\gamma = \frac{y\sin\frac{\phi}{2} - x\cos\frac{\phi}{2}}{q} \\ l+\cos\gamma = y\cos\frac{\phi}{2} + x\sin\frac{\phi}{2} \end{cases} \tag{4}$$

If we solve this system for $\cos\gamma$ and $\sin\gamma$, and apply the identity $\cos^2\gamma + \sin^2\gamma = 1$, we can get an implicit form of (x, y):

[4]Very likely the clamp position has to change. Therefore we focus on the re-usability of the 3 locators. Later we will discuss how to find the new clamp position.

$$ax^2 + 2bxy + cy^2 + d = 0 \quad (5)$$

where

$$\begin{cases} a = q^2 - pq\frac{\sin(\frac{\phi}{2}-\theta)}{\sin\frac{\phi}{2}} + \frac{p^2}{2-2\cos\phi} \\ b = \frac{2pq\cos\theta - p^2}{2\sin\phi} \\ c = q^2 - pq\frac{\cos(\frac{\phi}{2}-\theta)}{\cos(\frac{\phi}{2})} + \frac{p^2}{2+2\cos\phi} \\ d = -q^2(q - p\frac{\sin(\phi-\theta)}{\sin\phi})^2 \end{cases} \quad (6)$$

We can verify that the quadratic curve invariant $\delta = ac - b^2 > 0$. Therefore it is an elliptic curve. Since each edge is finite, we actually have an elliptic curve segment, where its end point will be determined by the endpoint of each edge (Figure 6). There are two possible curves of this kind. One is shown in figure 4. The other is shown in figure 5, and its equation can be derived similarly. Jia and Erdmann [7] apply this curve for geometric sensing of known planar shapes. The detail of a similar derivation of the elliptic curve can be found in [7].

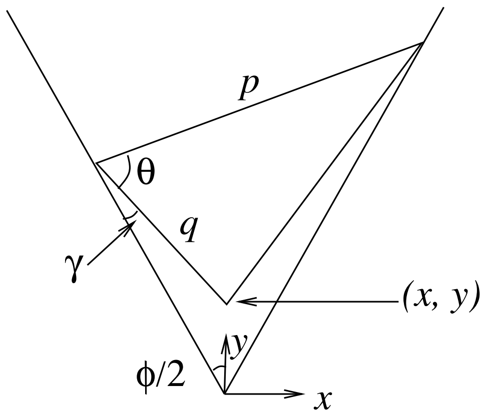

Figure 5: Another case of the elliptic curve can be derived similarly.

We call this curve the *contact locus* for design. In section 4.2 we will show the importance of this curve in the light of design rule.

4.2 The Design Rule for the Third Edge

The design rule for the third edge to guarantee the three-point contact with the same set of three locators is as following: *The third edge has to intersect the elliptic curve.* (Figure 6)

Indeed the design rule allows much bigger change of the part than that: you can change the part as much as you want, as long as the *boundary* of the new design intersects the contact locus

Note that this design rule gives an sufficient and necessary condition for 3-point contact. But it only gives

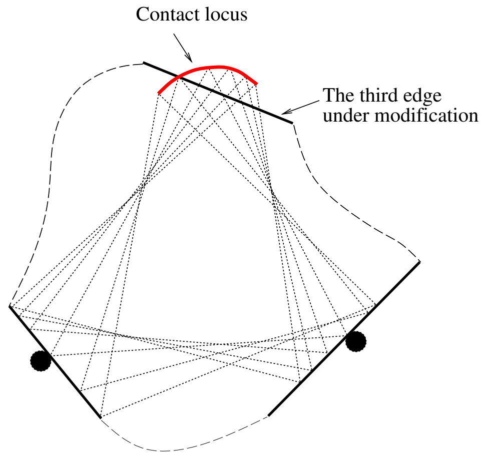

Figure 6: The third point contact can always be achieved as long as the third edge e_3 intersects the contact locus.

a necessary condition for form closure. We need to test the existence of clamp location for form closure.

5 Geometric Uncertainty

If the given part has bounded geometric uncertainty, we have to ask whether the given fixture still works. We follow [21, 22, 8] to characterize the geometric uncertainty using a *tolerance zone.* (Figure 7) By bounded geometric uncertainty, we mean there is a bounded tolerance zone associated to the edge in consideration (Figure 7). An edge can be anywhere within the tolerance zone, as long as it is fully contained in the zone. We can apply the contact locus to solve a restricted case of this problem.

If we assume that two of the three given edges are precise, while the third edge e_3 has bounded geometric uncertainty, we have the following sufficient condition for geometric tolerance: if the contact locus intersects the tolerance zone and cuts the zone into sufficiently small connected regions, then the part is guaranteed to be able to achieve 3 point contact with the 3 given locators. By sufficiently small connected regions, we mean that no single region is large enough to contain the entire edge. (Figure 7)

Since we are considering polygonal parts only, the tolerance zone of each edge is a rectangle. Therefore the intersection with the contact locus can be easily computed. The maximum width of each connected region is bounded by the maximum distance between all pairs of its vertices. If the width of each region is less than the minimum possible length of the edge, there always exists a 2D rigid body transformation of the part to achieve three point contact with the three given locators.

Given n fixtures, it is possible to rank them in terms

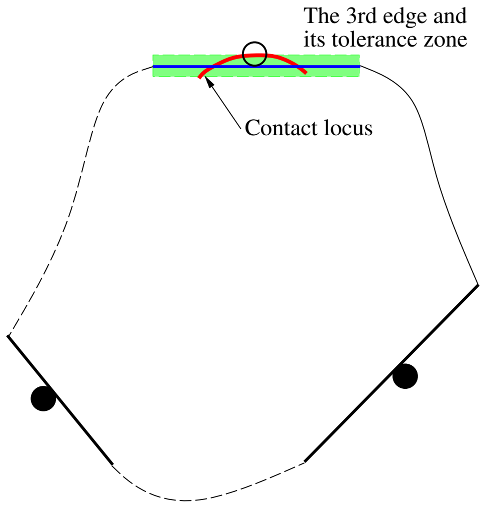

Figure 7: The part can always achieve three point contact, if the the tolerance zone is cut by the contact locus into sufficiently small connected regions.

of tolerance to uncertainty, using the contact locus. The width of the largest tolerance zone that can be cut into sufficiently small regions by the contact locus gives a numerical rank of the tolerance to uncertainty: the wider it is, the better it tolerates uncertainty. For example, if the part shown in figure 7 is modified by rotating its third edge as shown in figure 8, it has more tolerance to uncertainty because it has a wider tolerance zone.

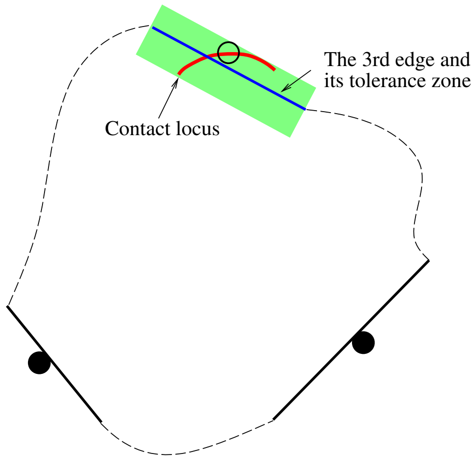

Figure 8: The rotated edge allows a wider tolerance zone, therefore the modified part tolerates more uncertainty.

6 The Clamp Location

So far, we are only concerned with achieving three point contact, which is necessary to re-use the fixture. A true fixture requires form closure. Therefore after achieving the three point contact with the three given locators, we have to find the clamp location to achieve form closure. We can apply the technique described in [2] to find all the possible clamp locations.

For each clamp location, we can verify the form closure condition by applying one of the following:

1. Each contact point defines a wrench vector. We have 4 such vectors. We can express one of them as a linear combination of the other three. Solve the linear system in $\mathcal{R}^3$. It is a form closure fixture if and only if all coefficient are positive.

2. Compute the convex hull of the tip of the 4 wrench vectors and the origin. It is a form closure if and only if the origin is not a vertex of the convex hull.

7 Discussion and Future Work

Manufacturing and installation of fixtures are costly for high-volume production runs. In this paper we address the problem of the *re-usability* of fixtures.

First we apply the geometric hashing technique to efficiently find multi-purpose fixtures, given a set of distinct parts and their corresponding set of fixtures. This algorithm is not restricted to polygonal parts. It applies to any shape of planar parts, given the set of locator triplets.

Next we present the design rule of planar parts, using the contact locus derived in section 4.1. This design rule can be incorporated into the CAD design cycle to rapidly check whether a proposed redesign is consistent with the given previous fixture. It allows us to integrate the design of product and the design of fixturing, and detect a potential fixture failure early. Zhuang and Goldberg [29] shows that there exists planar parts that cannot achieve 3 point contacts with any set of 3 locators. The contact locus presented in section 4.1 can guide the designer to modify such a part such that it is fixturable.

Finally in section 5 we show that we can apply the same contact locus to solve a restricted problem of fixturing under uncertainty. We are currently working on the general case that all edges lie within some tolerance zone. We believe that results in this area will be essential to MEMS fixturing.

Note that the design rule derived from the *contact locus* only gives a necessary condition for fixturing. Namely if the design rule is violated, a failure is detected immediately. However if the design rule is followed, we still have to check whether we can successfully find a clamp location for form closure. In our future work, we are going to investigate the possibility to integrate the clamp consideration into design rules.

The idea of contact locus can be extended to fixturing using V-shape edge fixels. In this case, the contact locus is a quartic curve. The intersection can still be solved in general, though more complicated. We are currently working on this.

The contact locus and design rule enable us to design an interactive user-interface. When the designer modifies the part, he or she can be instantly notified when the design rule is violated. We are currently implementing such an interface on WWW using Java applets. An interactive user-interface via WWW will enable users to share models and the software via Internet.

Acknowledgments

Thanks to John Canny for reviewing this paper and giving us valuable suggestions.

References

[1] H. Asada and Andre B. By. Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. *IEEE Journal of Robotics and Automation*, RA-1(2):86–94, June 1985.

[2] Randy C. Brost and Kenneth Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Transactions on Robotics and Automation*, 12(1):31–46, February 1996.

[3] Randy C. Brost and Ralph R. Peters. Automatic design of 3-d fixtures and assembly pallets. In *IEEE International Conference on Robotics and Automation*, April 1996.

[4] D. Fischer, R. Norel, R. Nussinov, and H.J. Wolfson. 3-d docking of protein molecules. In *Combinatorial Pattern Matching. 4th Annual Symposium*, June 1993.

[5] A. J. Goldman and A. W. Tucker. *Polyhedral Convex Cones*. Princeton University Press, Princeton, NJ., 1956.

[6] E. G. Hoffman. *Modular Fixturing*. Manufacturing Technology Press, Lake Geneva, Wisconsin, 1987.

[7] Yan-Bin Jia and Michael Erdmann. Geometric sensing of known planar shapes. *The International Journal of Robotics Research*, 15(4):365–392, August 1996.

[8] Leo Joskowicz, Elisha Sacks, and Vijay Srinivasan. Kinumatic tolerance analysis. *Second Workshop on Algorithmic Foundation of Robotics*, July 1996.

[9] Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, 1973.

[10] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 238–249, December 1988.

[11] Y. Lamdan and H.J. Wolfson. On the error analysis of geometric hashing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1991.

[12] Jean-Claude Latombe and Randall H. Wilson. Assembly sequencing with toleranced parts. In *Third Symposium on Solid Modeling and Applications*, 1995.

[13] Xanthippi Markenscoff, Luqun Ni, and Christos H. Papadimitriou. The geometry of grasping. *IJRR*, 9(1):61–74, February 1990.

[14] Bud Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(4):541–558, 1987.

[15] Van-Duc Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, 1988.

[16] Raquel Norel, Daniel Fisher, Haim J. Wolfson, and Ruth Nussinov. Molecular surface recognition by a computer vision-based technuque. *Protein Engineering*, 7(1):39–46, 1994.

[17] E. N. Ohwovoriole. Kinematics and friction in grasping by robotic hands. *Transactions of the ASME*, 109, 1987.

[18] Mark Overmars, Anil Rao, Otfried Schwarzkopf, and Chantal Wentink. Immobolizing polygons against a wall. In *ACM Symposium on Computational Geometry*, 1995. Vancouver, BC.

[19] Kamen Penev and Aristides A. G. Requicha. Fixture foolproofing for polygonal parts. In *International Symposium on Assembly and Task Planning*. IEEE, August 1995. Pittsburgh, PA.

[20] Jocelyn Pertin-Troccaz. Grasping: A state of the art. In *The Robotics Review I*, pages 71–98. MIT Press, 1989. edited by O. Khatib, J. J. Craig, and T. Lozano-Perez.

[21] A. A. G. Requicha. Toward a theory of geometric tolerancing. *IJRR*, 2(4), 1983.

[22] A. A. G. Requicha. Mathematical definition of tolerance specifications. *Manufacturing Review*, 6(4), December 1993.

[23] F. Reuleaux. *The Kinematics of Machinery*. Macmilly and Company, 1876. Republished by Dover in 1963.

[24] 2nd Edition VLSI Technology, editor. *S. M. Sze*. McGraw-Hill Publishing Company, 1988.

[25] Aaron Wallack and John Canny. Planning for modular and hybrid fixtures. In *International Conference on Robotics and Automation*. IEEE, May 1994.

[26] Chantal Wentink, A. Frank van der Stappen, and Mark Overmars. Fixture planning. In Jean-Paul Laumond and Mark Overmars, editors, *Workshop on Algorithmic Foundations of Robotics*, July 1996.

[27] H.J. Wolfson. Model-based object recognition by geometric hashing. In *Computer Vision. First European Conference on Computer Vision*, pages 526–536, April 1990.

[28] Chee Yap. Geometric tolerancing: Theory, issues, and computation. *Second Workshop on Algorithmic Foundation of Robotics*, July 1996.

[29] Yan Zhuang and Ken Goldberg. On the existence of solutions in modular fixturing. *The International Journal of Robotics Research*, 15(6):646–656, December 1996.