Computing Tolerance Parameters for Fixturing and Feeding

Jingliang Chen, University of California at Berkeley, USA
Ken Goldberg[*], University of California at Berkeley, USA
Mark H. Overmars, University of Utrecht, The Netherlands
Dan Halperin, Tel Aviv University, Israel.
Karl F Böhringer, University of Washington, USA
Yan Zhuang, Qualcomm Inc., USA

1 January 2002

Abstract

Fixtures and feeders are important components of automated assembly systems: fixtures accurately hold parts and feeders move parts into alignment. These components can fail when part shape varies. Parametric tolerance classes specify how much variation is allowable. In this paper we consider fixturing convex polygonal parts using right-angle brackets and feeding polygonal parts on conveyor belts using sequences of vertical fences. For both cases, we define new tolerance classes and give algorithms for computing the parameter specifications such that the fixture or feeder will work for all parts in the tolerance class. For fixturing we give an $O(1)$ algorithm to compute the dimensions of rectangular tolerance zones. For feeding we give an $O(n^2)$ algorithm to compute the radius of the largest allowable tolerance zone around each vertex. For each, we give an $O(n)$ time algorithm for testing if an n-sided part is in the tolerance class.

[*] goldberg@ieor.berkeley.edu

1 Introduction

Issues in tolerancing and precision date back to the Industrial Revolution. *Variation* describes how parts differ and *tolerance* is the amount of variation that can be tolerated. The goal of tolerancing is "to define a class of objects that are (1) interchangeable in assembly operations and (2) functionally equivalent" [23]. During assembly, variation can cause processes such as fixturing and feeding to fail, resulting in damage and costly production delays.

In 1994, following a dozen years of preparation, the American Society of Mechanical Engineers published the ASME Y14.5M standard for dimensioning and tolerancing along with ASME Y14.5.1M, a set of mathematical definitions of concepts such as a planar datum surface [19, 28]. That definition is based on an uncountable set of lines in the plane; computational definitions remain elusive.

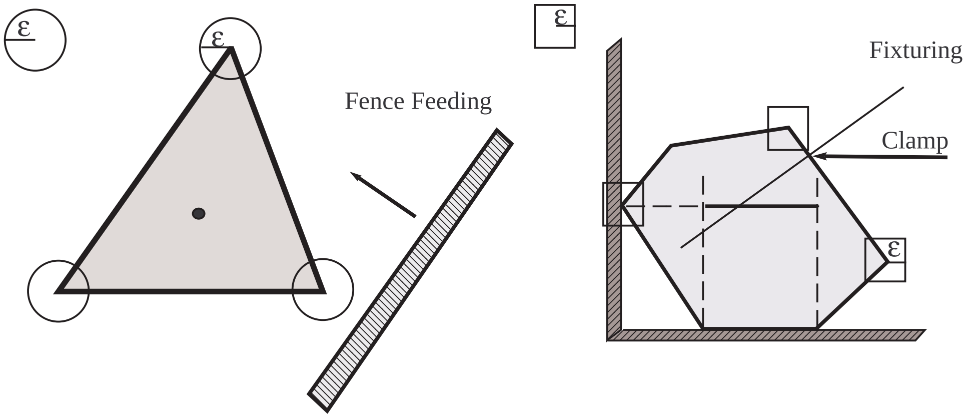

Figure 1: Tolerance classes for feeding and fixturing based on part geometry. On the left, the tolerance parameter ε defines the radius of a disk that specifies allowable variation in the location of the vertex for fence-based part feeders. On the right, the tolerance parameter is the width of a square box around each vertex that specifies allowable vertex variation for a given part fixtured with a right-angle bracket.

In this paper we define new tolerance classes and algorithms for computing their parameters. We build on recent models for fixturing and feeding that assume nominal part geometry as input. Given the nominal part P, it is possible to automatically compute a fixture or a set of feeding angles. As illustrated in Figure 1, we define tolerance classes $\Delta(P, \varepsilon)$ based on the nominal part and a variation parameter ε. We can check if a given part is in a class in time linear in the number of part vertices. The parameter ε is chosen so that the fixture or feeder will work for all parts in the tolerance class.

2 Related Work

Zone-based tolerance models require only that the part fit within a defined geometric zone: since part geometry can be arbitrarily baroque while remaining within the zone, it is extremely difficult to characterize the range of resulting part behaviors. In contrast, parametric tolerance models use a finite number of parameters to define a range of allowable shapes. One scalar parameter might indicate the maximal distance that an edge defined by a datum surface can vary from its ideal (nominal) position. In parametric models, deviations from linearity along an edge are considered negligable.

Requicha's review of industrial tolerancing practice highlighted the role of "perfect form" datum surfaces: perfect lines or arcs that a matching surface of the imperfect part must rest on prior to measurement. Requicha noted a paucity of computational definitions in tolerancing and proposed a formal definition of the popular maximal material conditions (MMC) and least material conditions (LMC) based on the distance between a point to a subset of Euclidean space (E^3) and the regularized difference and complementarity operators that can precisely specify positive and negative offset solids. Requicha's paper closed with a call for efficient algorithms to compute tolerance specifications [23]. Other approaches to computer aided tolerancing can be found in [5, 24, 31, 26, 27].

Yap and Chang [32] consider geometric metrology: given a set of sample points on the boundary of a closed curve, decide if the curve lies within a given geometric tolerance zone. Even for a (one-dimensional) line segment, conservative classification policies may not exist; the authors illustrate that deterministic policies and statistical decision rules can be non-trivial even in one dimension. We define parametric tolerance models that permit exact classification in linear time.

Brooks [7] and Donald [11] consider shape variation for robot motion planning. Brooks shows how parametric variables can be propagated through an assembly and Donald treats parametric variables as additional dimensions in configuration space. Hutchinson and Kac developed an AI-based approach to tolerancing for assembly [13]. Latombe, Wilson and Cazals also consider tolerancing in the context of assembly [17]. They propose a parametric tolerance model similar to ours in that they assume part edges are straight. To make the analysis manageable, they add the requirement that edges/faces maintain their relative orientation: i.e., edges in the tolerance class always remain parallel to the nominal edge. Given a parametric tolerance class, the authors give polynomial time algorithms to decide if an assembly sequence exists that is guaranteed for all parts in the class. Computing parametric tolerance bounds for mechanisms using kinematic pairs is the subject of an ongoing analytic project by Joskowicz and Sacks [14].

Brost and Peters [8] propose a tolerance class for modular fixturing where contact normals are allowed to vary within a cone of a given half angle. Kavraki considered variation in part shape in the context of orienting parts with a planar force field. She was able to bound the final orientation of a part in terms of the shape difference between parts P and P': $P - P' \cup P' - P$ [15, 16].

The work closest to our feeding model is by Akella and Mason [1, 3]. They also consider fence plans for feeding polygonal parts and parametric tolerance classes based on circles around part vertices. One difference is that they define a circular tolerance zone around the part's center of mass. They note that determining a bound on the possible center of mass of a uniform mass polygon, in terms of bounds on its vertices, remains an open problem. Instead, we define a tolerance class relative to a coordinate frame centered at the part's center of mass. Given the radius of a circular tolerance zone around each vertex and the center of mass, Akella and Mason compute action ranges and use breadth-first search of an AND/OR tree to check if a solution exists. We modify their model for shape tolerance and give new algorithms for computing its parameters. Other references specific to fixturing and feeding are described below. A preliminary version of these results appeared in [9].

3 Parametric Tolerance Classes for Fixturing

A fixture is a device that locates and holds parts during assembly, inspection, or machining. Modular fixturing systems typically include a lattice of holes with precise spacing and an assortment of precision locating and clamping modules that can be rigidly attached to the lattice. An excellent survey of recent research can be found in [29].

Let P denote a convex polygonal projection of a part without parallel edges. As illustrated in Figure 2, such a part can be fixtured using one right-angle bracket and one clamp. Wentink, Van der Stappen and Overmars proved that any convex polygonal part can be fixtured in this manner [20, 25]. We make the conservative assumption that the contacts are frictionless: a fixture computed assuming zero friction will also immobilize the part when friction is non-zero.

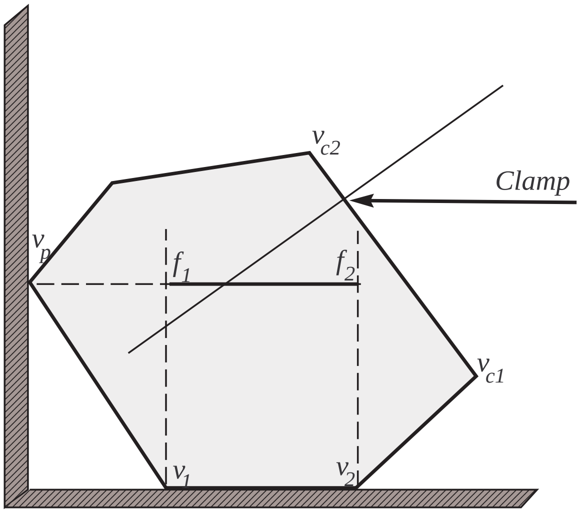

Figure 2: The part P (shaded) in a fixture with right angle bracket and clamp point.

When P is placed into the fixture, it slides into place on the horizontal arm of the bracket with an edge contact and also makes a point contact on the vertical arm of the bracket. Moving from the right to left, the clamp establishes a point contact with a remaining edge of the part which we refer to as the *clamp edge* of the part.

We label P's vertices counterclockwise: $v_1, \ldots, v_n$. We label the two endpoints of the horizontal edge contact v_1 and v_2, the point contact on the vertical arm v_p, and the endpoints of the clamp edge as v_{c1} and v_{c2}. We assign a coordinate frame to the part; its origin, c, is at midpoint of the edge v_1v_2 and we align its x-axis with the edge v_1v_2.

If we draw vertical lines from v_1 and v_2, they intersect a horizontal line from v_p at points f_1 and f_2 respectively. We refer to the line segment f_1f_2 as the *fixturing segment*. The right angle bracket and clamp will immobilize the part if the normal of the clamp edge at the clamp contact point intersects the fixturing segment. For details see [20].

We define the parametric tolerance class $\Delta(P, \varepsilon)$, assuming that part edges are perfectly linear. This class specifies allowable variation in the location of part vertices using square boxes and line segments.

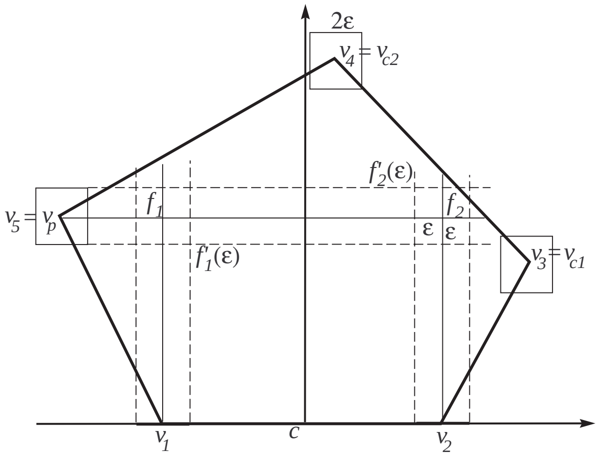

Figure 3: Parametric tolerance class based on parameter ε.

As illustrated in Figure 3, coordinates are expressed relative to the part coordinate frame. Let P' be a candidate part. P' is loaded into the fixture by placing it with edge $v_1'v_2'$ on the horizontal bracket edge, shifting the part to the left until it hits the vertical bracket edge, and then moving the clamp horizontally until it touches the part. Let c' denote the center of the edge $v_1'v_2'$. We say that $P' \in \Delta(P, \varepsilon)$ if we can place part P' such that c' is coincident with c, the bottom edges are aligned, and all the vertices of P' are within a square of size ε around the corresponding vertices of the nominal part P. It is easy to check if a candidate $P' \in \Delta(P, \varepsilon)$ in $O(n)$ time.

When the part shape changes in the tolerance class, the fixturing segment may change in height and length, the clamp contact point may move horizontally and its normal may rotate. Also, the vertex v_p that makes the contact with the vertical arm of the bracket may be changed. In this paper, we restrict our tolerance zone such that this last type of change does not happen and we only consider the case where v_1, v_2, v_p, v_{c1}, and v_{c2} are distinct vertices.

3.1 Computing ε for Fixturing

Since we assume that v_1,v_2,v_p,v_{c1} and v_{c2} are distinct vertices, a fixture is valid for $\Delta(P, \varepsilon)$ if and only if for every $P' \in \Delta(P, \varepsilon)$, the normal at the clamp lies between $f_1(\varepsilon)$ and $f_2(\varepsilon)$.

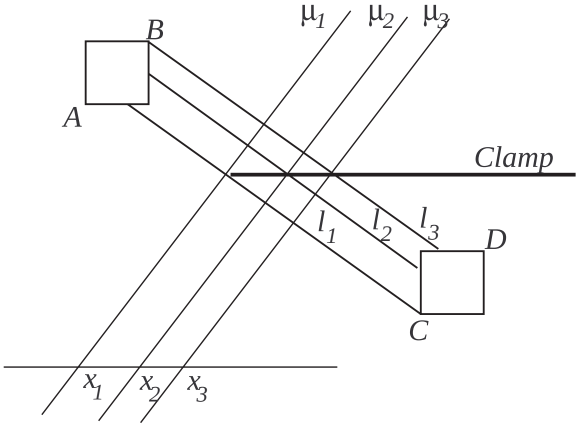

Figure 4: The boundary condition for fixturing.

Consider the clamp edge of the part. The normal at the clamp point will intersect the critical fixture segment at some position x. The part edge that achieves the maximum (or minimum) x must have one endpoint at one of the four extreme points ($ABCD$) since we could move the edge parallel to itself until one of the endpoints reaches the boundary, and get a larger (smaller) x.

Now we fix one of the endpoints of the part edge and look at the relationship between the orientation of the clamp edge and the x-coordinate of intersection of the normal and the fixture segment. As indicated in Figure 5, let y be the vertical distance from the fixed endpoint to the fixture segment, and let h be the vertical distance from the clamp to the fixture segment. Let x_1 be the x-coordinate of the fixed endpoint, x_2 the x-coordinate of the clamp contact point, and x_3 the x-coordinate of intersection between the normal and the fixture segment. Let θ be the orientation of the clamp edge. Because the normal is perpendicular to the clamp edge, we have

$$\frac{y-h}{x_2-x_2} = \frac{x_2-x_3}{h} = \tan\theta \tag{1}$$

From Equation 1 , we eliminate x_2 and get

$$x_3 = x_1 + (y-h)\cot\theta - h\tan\theta \tag{2}$$

Taking the derivative of x_3 with respect to θ and setting it equal to 0,

$$-\frac{y-h}{\sin^2\theta} - \frac{h}{\cos^2\theta} = 0 \tag{3}$$

Solving for θ,

$$\theta = \arctan\sqrt{1-\frac{y}{h}} \tag{4}$$

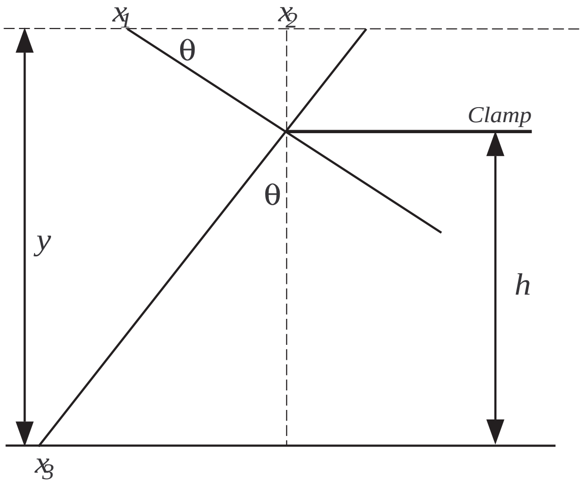

Figure 5: Relationship between θ and x_3 when we fix one endpoint of the clamp edge.

Equation 4 has at most one solution in $[0, \pi/2]$. So, given ε, for each fixed endpoint we only need to consider at most three directions: the maximal and minimal θ within the tolerance zone and the solution of Equation 4. Also we only need to consider four fixed endpoints (the corners of the tolerance box) at each side of the clamp edge. Looking at this carefully, we get a total of 8 clamp edges and 2 points ($f_1(\varepsilon)$ and $f_2(\varepsilon)$) within the tolerance zones that might define the maximum or minimum position of the intersection of the normal and the fixture edge. (The 8 edges are the four tangent lines of the two tolerance boxes plus, for each of the 4 extreme vertices (2 per box), the angle that comes out of Equation 4.) This leads to a system with a constant number of equations from which we can compute the largest ε in which the fixture design will work for all parts in $\Delta(P, \varepsilon)$ (assuming no topological change). Since only five vertices are relevant in the calculation of ε, the complexity of this algorithm is $O(1)$. For details see [9]. We implemented this algorithm and computed ε for the parts and fixtures shown in Figure 6.

4 Parametric Tolerance Classes for Part Feeding

A part feeder is a device that accepts parts in random orientation and outputs them in a unique orientation. The design of part feeders is still a "black art" that is responsible for up to 30% of the cost and 50% of workcell failures [6]. An algorithm for feeding polygonal parts was proposed in [12] and subsequently refined in a series of papers [22, 10, 4]. The model we consider here orients parts of a given shape as they are pushed by a sequence of fences on a conveyor belt [21, 2, 30, 4]. A solution is a sequence of pushing angles.

Using the notation of [4], let P be the nominal convex polygonal part with n vertices and center of mass c. We assign a fixed coordinate frame to P, with c as its origin and x-axis aligned with the desired nominal final orientation of the part after feeding.

As illustrated in Figure 7, all angles are expressed relative to this coordinate frame. Vertices are counterclockwise labeled $v_1 \ldots v_n$ and e_i is the edge connecting v_i and v_{i+1}; the edge e_n connects v_n and v_1. Let α_{v_i} be the direction of the ray cv_i in the coordinate system and α_{e_i} be the direction of the normal from c to the line containing the edge e_i. Let $\alpha_{r_i} = \alpha_{e_i} - \alpha_{v_i}$,

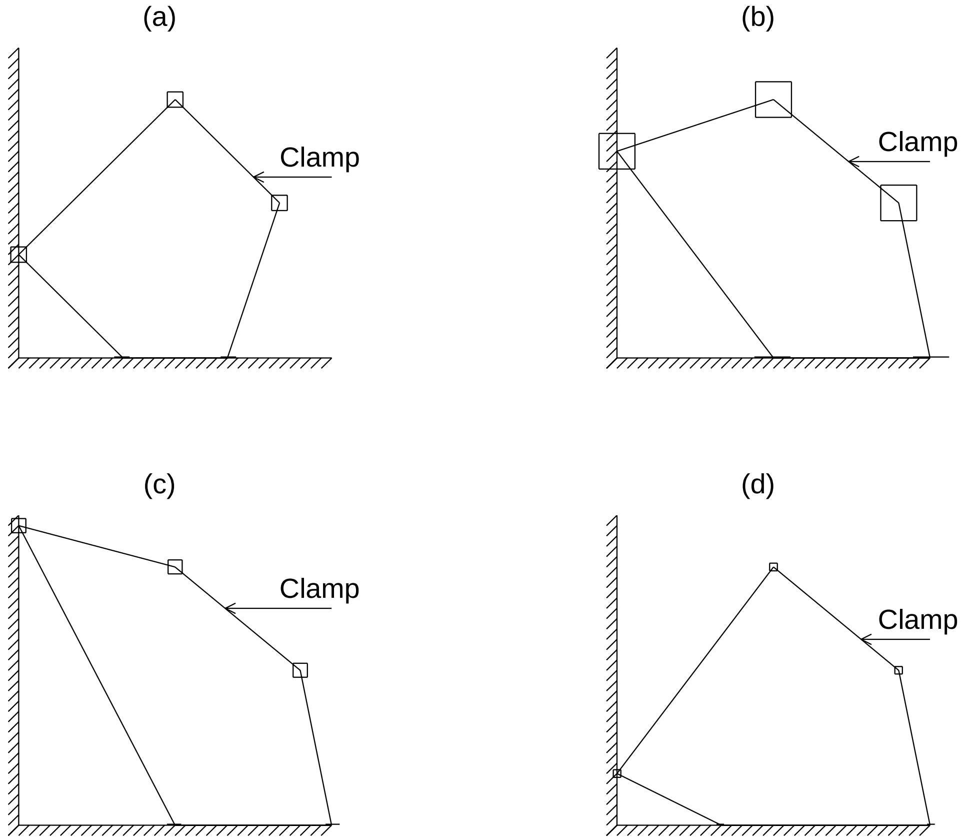

Figure 6: Four parts and fixtures with computed tolerance parameters.

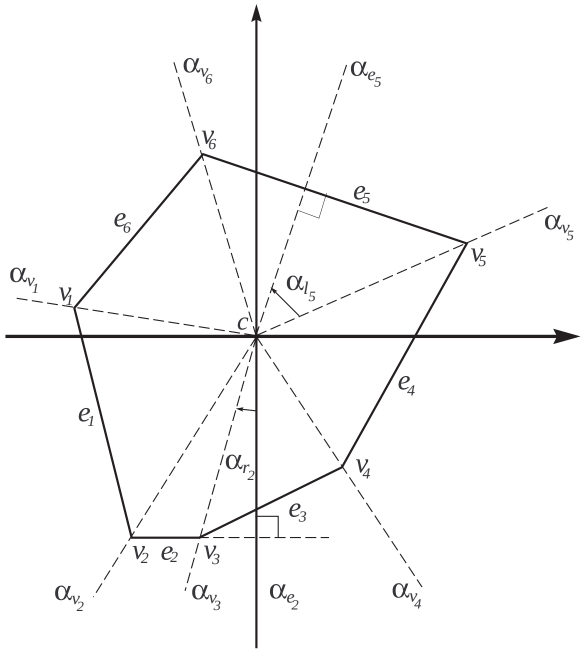

Figure 7: Critical part angles for feeding as defined from the part's center of mass.

$\alpha_{l_i} = \alpha_{v_{i+1}} - \alpha_{e_i}$.

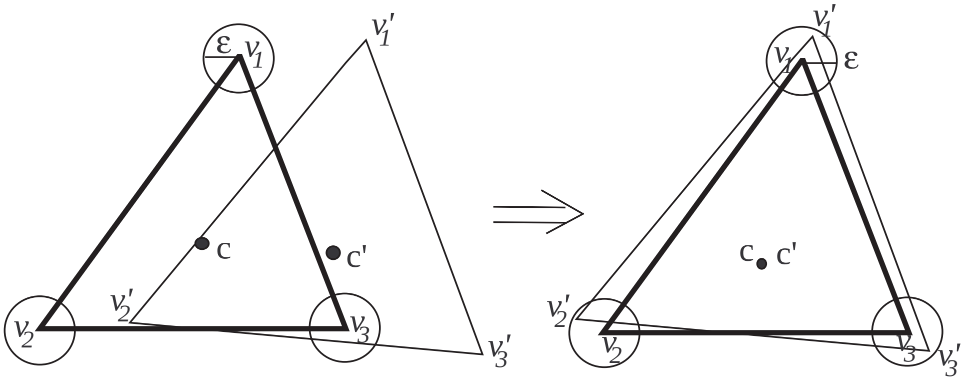

Figure 8: Parametric tolerance class $\Delta(P, \varepsilon)$ for feeding and procedure to check if a candidate part P' is in the class.

Figure 8 illustrates the parametric tolerance class $\Delta(P, \varepsilon)$ for part feeding based on a nominal part P with center of mass c and a test to check if a candidate part P' is in the class. We assume that part edges are always straight. Each vertex of P has a circular tolerance zone of radius ε. A part P' with center of mass c' is in $\Delta(P, \varepsilon)$ if we can position and orient P' such that c' is aligned with c and all of P''s vertices are inside the tolerance zones around the corresponding vertices of P. We assume that topologies of the convex hull of all the parts in $\Delta(P, \varepsilon)$ are the same. We can test whether or not a polygon P' is in $\Delta(P, \varepsilon)$ by computing an angular interval Ω_i' for each vertex v_i' such that rotating P' by $\omega \in \Omega_{i'}$ will move v_i' into the ε disk around v_i, or will keep it inside the disk if it is already inside the disk. If $\cup_i \Omega_i' \neq \emptyset$, then $P' \in \Delta(P, \varepsilon)$. The complexity of this test is $O(n)$.

4.1 The Push Function and Push Plans

As shown in Figure 9, the *push function* characterizes the mechanics of a part pushed by a frictionless planar fence [18, 12]. The push function $p : [0, 2\pi) \rightarrow [0, 2\pi)$ links every orientation ϕ to the orientation $p(\phi)$ in which the part P settles after being pushed by a jaw with initial push direction ϕ.

The push function p of a polygonal part consists of steps, which are intervals $I \subset [0, 2\pi)$ such that for all $\phi \in I$, $p(\phi) = C$ for some constant $C \in$

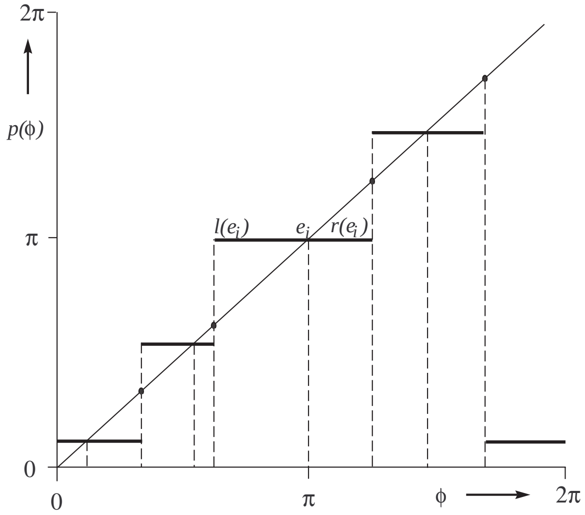

Figure 9: The push function for a polygonal part. Each step corresponds to a stable edge of the part.

I; see Figure 9. The steps of the push function are easily constructed from the α_{l_i}'s and α_{r_i}'s. The interval I is bounded by two consecutive unstable equilibrium push directions. All directions strictly inside the interval map onto the stable equilibrium push directions corresponding to a stable edge. (Note that the direction C itself maps onto C because it is the direction of the stable edge.) As a result, the number of steps in the push function equals the number of stable edges.

In preparation for the next section, we define for each stable edge e_i two open intervals $l(e_i) = \{\phi < \alpha_{e_i} | p(\phi) = \alpha_{e_i}\}$ and $r(e_i) = \{\phi > \alpha_{e_i} | p(\phi) = \alpha_{e_i}\}$. We refer to these intervals as e_i's left and right environment respectively. The interval $l(e_i)$ corresponds to the half-step left of α_{e_i} and $r(e_i)$ corresponds to the half-step right of α_{e_i} (Figure 9). The open intervals $l(e_i)$ and $r(e_i)$ are maximal intervals without equilibrium orientation.

We use the abbreviation p_α to denote the (shifted) push function defined by

$$p_\alpha(\phi) = p((\phi + \alpha) \bmod 2\pi)$$

for all $\phi \in [0, 2\pi)$. Note that $p_\alpha(\phi)$ is the final orientation of a part in initial orientation ϕ after a reorientation by α followed by a push. We can now define a *push plan*.

Definition 3.1 *A push plan is a sequence* $\alpha_1, \ldots, \alpha_m$ *such that* $p_{\alpha_m} \circ \ldots \circ p_{\alpha_1}(\phi) = \Phi$ *for all* $\phi \in [0, 2\pi)$ *and some fixed* $\Phi \in [0, 2\pi)$.

Goldberg [12] showed that a push plan exists for any polygonal part and gave an $O(n^2)$ algorithm for finding the shortest such plan. Chen and Ierardi [10] gave a different algorithm based on finding a basic pushing action that repeatedly applies one critical pushing angle to orient the part. The critical pushing angle depends on the length of the two largest open left half intervals in the push function. The same is true for right intervals but, without loss of generality, we only consider left push plans here. Let Ψ be a range of angles defined by the difference in length of these half intervals. Any angle in Ψ will serve as the critical pushing angle. This range can compensate for variation in part shape.

Consider the open intervals $l(e_i)$ defined in the previous section. Let α be the length of the longest interval in the set. We assume, without loss of generality, that e_1 in P is a stable edge with $|l(e_1)| = \alpha$, and we only consider the case in which P has no second stable edge e_i $(i \neq 1)$ satisfying $|l(e_i)| = \alpha$. Each basic action consists of a reorientation of the jaw by an angle of $-(\alpha - \mu)$, with $\mu > 0$ such that $\alpha - \mu > |l(e_i)|$ for any stable edge $e_i \neq e_1$, and a subsequent application of the jaw. (The same arguments will apply for $r(e_i)$ with reorientation by an angle of $\alpha - \mu$ instead of $-(\alpha - \mu)$.)

Note that a reorientation of the jaw by $-(\alpha - \mu)$ corresponds to a change of the push direction by $\alpha - \mu$. Every basic action puts the part into a stable equilibrium orientation. If P is a polygonal part, then the stable equilibrium orientations occur at isolated points in $[0, 2\pi)$. After each basic action, the number of possible part orientations is finite. Let us label the m stable edges $e_{s_1}, \ldots, e_{s_m}$ in order of *increasing* edge index with $s_1 = 1$. After the first application of the jaw, the part P can be in the orientation of any stable edge $e_{s_1}, \ldots, e_{s_m}$. Chen and Ierardi show that every next basic action eliminates the last stable edge in the sequence. So an $m + 1$ step push plan $p_{\alpha-\mu} \circ \ldots \circ p_{\alpha-\mu}$ suffices to put P into orientation α_{e_1}.

4.2 Effect of Shape Tolerance on the Push Function

When part geometry changes, it affects the push function in two ways. First, it will change the value of α_{l_i} and α_{r_i}, that is, it will change the length of one or more open half intervals. Second, some critical change in the value of α_{l_i} and α_{r_i} may cause a stable edge in the nominal part to become unstable.

Also, it may cause an unstable edge to become stable. Since the number of steps of the push function is equal to the number of stable edges, this will combine several half intervals into a bigger one or break one half interval into several smaller ones. We refer to this as a *topological change* of the push function.

Using Chen and Ierardi's algorithm, we can find a push plan that is valid for all the parts in the tolerance class $\Delta(P, \varepsilon)$ if

- for every part in $\Delta(P, \varepsilon)$, e_1 is the stable edge with the largest left half interval, and
- if each member $P' \in \Delta(P, \varepsilon)$ is characterized by a pushing angle set Ψ', and the intersection of all Ψ' is nonempty.

We can state this differently. If the lower bound of $|l(e_1')|$ for any part $P' \in \Delta(P, \varepsilon)$ is larger than the upper bound of all other left half intervals, then there will be a push plan which will work for all the parts in the tolerance class.

4.3 Computing ε for Feeding

To compute the tolerance parameter ε, the algorithm takes as input the geometry of the nominal part, computes a push angle α that defines the push plan, and a bound on ε such that there exists a plan to feed all the parts in $\Delta(P, \varepsilon)$.

We assume vertex v_i is an equilibrium vertex and that there is no topological change in the push function within the tolerance class $\Delta(P, \varepsilon)$. Non-equilibrium vertices and topological changes in the push function are treated in [9].

As shown in Figure 10, let $\beta_i = \pi/2 - \alpha_{l_i}$. To derive a lower bound on $l(e_i) = \alpha_{l_i}$, note that minimizing α_{l_i} is equivalent to maximizing β_i. As shown in Figure 10 and by definition $\alpha_{l_i} + \beta_i = \pi/2$. For every v_i' within the ε disk around v_i, the maximum β_i is achieved when e_i' is tangent to the ε disk around v_{i+1}. So given the vertex v_i', the maximum value of α among all $P' \in \Delta(P, \varepsilon)$ and $v_i' \in P'$ is well defined. The problem now becomes to find the point v_i' that can maximize β_i. For any v_i' *inside* the ε disk around v_i, there is a point v_i'' *on* the ε disk around v_i such that $\beta_i'' > \beta_i$. Thus we need only consider points on the boundary of the circular tolerance zone.

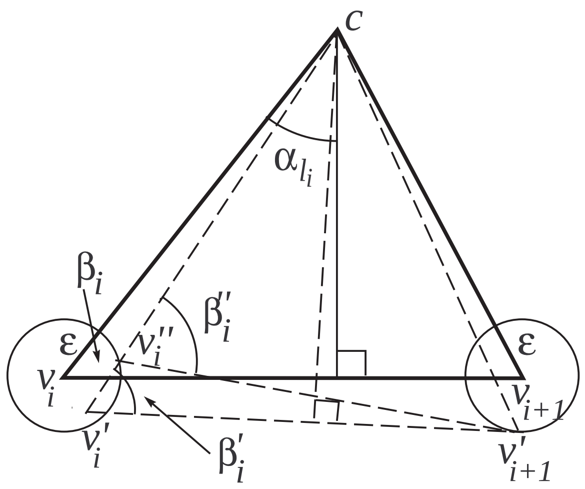

Figure 10: Construction for computing the tolerance parameter ε.

A point v_i' on the circle around v_i can be represented by a single variable θ. $v_i'v_{i+1}'$ is tangent to the ε disk around v_{i+1} since this maximizes β_i' when we fix v_i'. To derive the relationship between β_i' and ε, we draw a straight line $v_i'q$ parallel to v_ic with cq perpendicular to $v_i'q$ and $v_i'k$ parallel to v_iv_{i+1} with $v_{i+1}k$ perpendicular to $v_i'k$. d_i is the length of e_i and h_i is the distance from the center of mass to e_i. Let $\angle cv_i'q = \gamma$, $\angle kv_i'v_{i+1} = \lambda$, $\angle v_{i+1}v_i'v_{i+1}' = \delta$. We have:

$$\beta_i' = \gamma + \beta_i + \delta + \lambda \tag{5}$$

When we only consider points on the ε disk, the relationship between $\min(|l(e_i')|)$ and ε can be expressed [9] as

$$\begin{aligned} \min(|l(e_i')|) &= f_i(\varepsilon) \\ &= |l(e_i)| - \arcsin\frac{\varepsilon}{\sqrt{(d_i - \varepsilon\cos\theta)^2 + \varepsilon^2\sin^2\theta}} \\ &\quad - \arcsin\frac{\varepsilon\sin(\beta_i - \theta)}{\sqrt{(|cv_i| - \varepsilon\cos(\beta_i - \theta))^2 + \varepsilon^2\sin^2(\beta_i - \theta)}} \\ &\quad - \arcsin\frac{\varepsilon\sin\theta}{\sqrt{(d_i - \varepsilon\cos\theta)^2 + \varepsilon^2\sin^2\theta}} \end{aligned} \tag{6}$$

where θ is the value in $[0, 2\pi)$ that minimizes the right-hand side. A similar relationship holds for $\max(|l(e_i')|)$. For every stable edge e_i, if there is no topological change in the push function, we can express the relationship between $|l(e_i')|$ and ε using this set of nonlinear constraints. We can compute the largest ε with a binary search in the topological range over $[0, min(0.99h_i, 0.49d_i)i = 1 \ldots n]$. For each ε we search incrementally, testing $\theta \in [0, 2\pi]$ to find a value that yields $f_i(\varepsilon)$. Since we can not guarantee monotonicity of f_i in θ, it may be necessary to backtrack and choose smaller increments. Handling topological changes is discussed in [9], the overall complexity for computing ε is $O(n^2)$. We implemented the algorithm and show examples of the resulting tolerance classes in Figures 11 and 12.

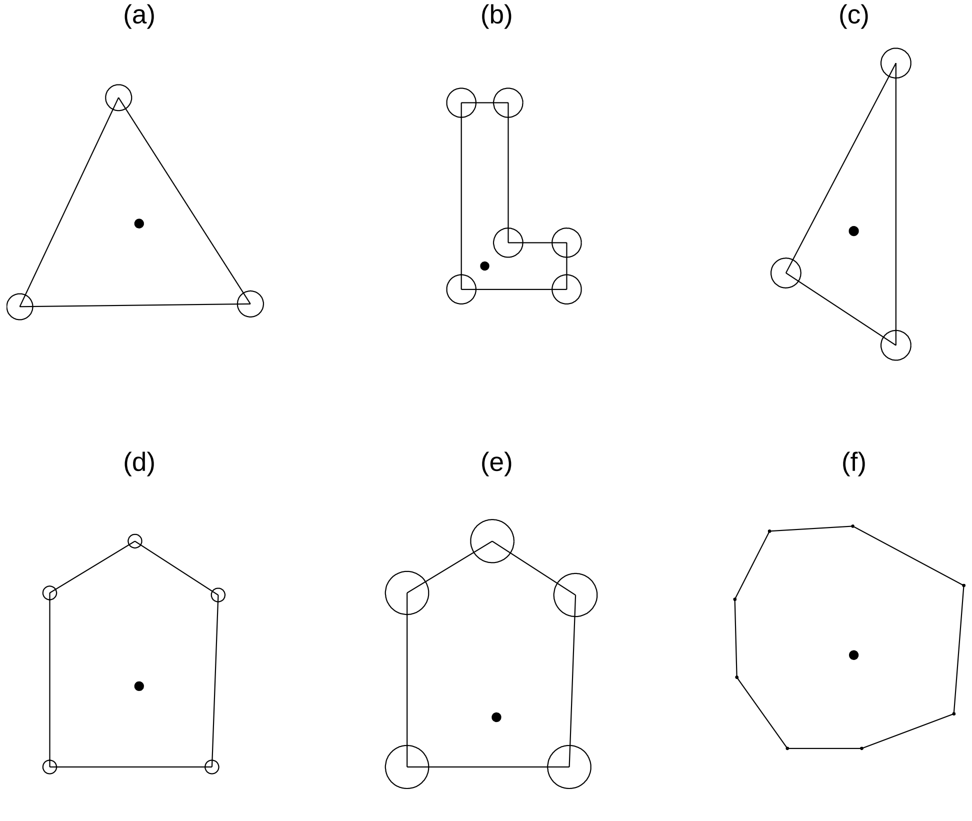

Figure 11: Tolerance classes for feeding: six example parts.

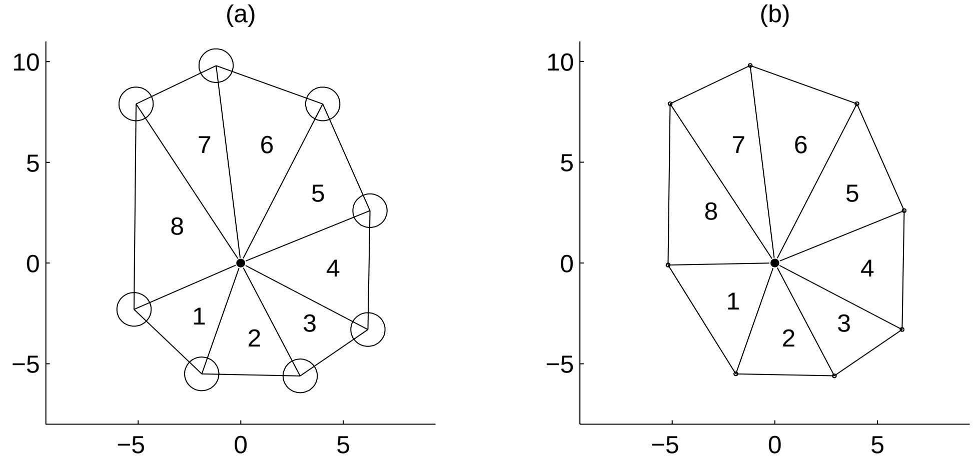

Figure 12: Small changes in polygon shape can produce large changes in tolerance parameters. The part on the left has a relatively large tolerance parameter while the one on the right has a much smaller tolerance parameter. This is due to a very small change in the position of the leftmost vertex. In this way the algorithm can reveal tolerancing implications that may not be obvious from visual inspection.

5 Conclusion

The widespread adoption of CAD/CAM for assembly has created a renewed interest in computational tolerancing and algorithms for computing parametric tolerance classes. Since assembly components such as fixtures and feeders can fail when part shape varies, it is important to formalize how part tolerance affects their behavior.

This paper contributes by rigorously characterizing two new parametric tolerance classes. For fixturing we give an $O(1)$ algorithm to compute the dimensions of rectangular tolerance zones. For feeding we give an $O(n^2)$ algorithm to compute the radius of the largest allowable tolerance zone around each vertex. For each, we give an $O(n)$ time algorithm for testing if an n-sided part is in the class. We implement both algorithms and illustrate with examples.

Acknowledgements

This work was supported in part by NSF Awards IRI-9612491 and Presidential Faculty Fellow Award IRI-9553197 to Prof. Goldberg and by NATO Travel Grant CRG 951224 (joint with M. Overmars).

Work by D.H. has been supported in part by the USA-Israel Binational Science Foundation, by the IST Programme of the EU (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

References

[1] S. Akella. *Robotic Manipulation for Parts Transfer and Orienting: Mechanics, Planning, and Shape Uncertainty.* PhD thesis, The Robotics Institute, Carnegie Mellon University, Dec. 1996. Robotics Institute Technical Report CMU-RI-TR-96-38.

[2] S. Akella, W. Huang, K. Lynch, and M. Mason. Planar manipulation on a conveyor by a one joint robot with and without sensing. In *Second Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.

[3] S. Akella and M. Mason. Orienting toleranced polygonal parts. *International Journal of Robotics Research*, 19(12), December 2000.

[4] R. P. Berrety, K. Goldberg, M. Overmars, and F. van der Stappen. Computing fence designs for orienting parts. *Computational Geometry, Theory, and Applications*, 10(4):249–262, 1998.

[5] O. Bjorke. *Computer Aided Tolerancing.* ASME Press, 1989.

[6] G. Boothroyd, C. Poli, and L. E. Murch. *Automatic Assembly.* Marcel Dekker, Inc., 1982.

[7] R. A. Brooks. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, 1(4):29–68, Winter 1982.

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

[9] J. Chen, K. Goldberg, M. Overmars, D. Halperin, K. Bohringer, and Y. Zhuang. Shape tolerance in feeding and fixturing. In *International Workshop on Algorithmic Foundations of Robotics*. 1998.

[10] Y.-B. Chen and D. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14, 1995.

[11] B. R. Donald. *Error Detection and Recovery in Robotics.* Springer-Verlag, 1987.

[12] K. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2):201–225, August 1993. Special Issue on Computational Robotics.

[13] S. A. Hutchinson and A. C. Kak. Extending the classical ai planning paradigm to robotic assembly planning. In *IEEE International Conference on Robotics and Automation*, 1990.

[14] L. Joskowicz, E. Sacks, and V. Srinivasan. Kinematic tolerance analysis. In *Second Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.

[15] L. Kavraki. Part orientation with programmable vector fields: Two stable equilibria for most parts. In *IEEE International Conference on Robotics and Automation (ICRA)*, Albuquerque, New Mexico, Apr. 1997.

[16] F. Lamiraux and L. Kavraki. Positioning of symmetric and nonsymmetric parts using radial and constant fields: computation of all equilibrium configurations. *Intl. Journal of Robotics Research*, 20(8):635–59, 2001.

[17] J. C. Latombe, R. H. Wilson, and F. Cazals. Assembly sequencing with toleranced parts. In *Third Symposium on Solid Modelling and Applications*, 1995.

[18] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3), Fall 1986.

[19] A. Neumann. The new ASME Y14.5M standard on dimensioning and tolerancing. *Manufacturing Review*, 7(1), 1994.

[20] M. Overmars, C. Wentink, and F. van der Stappen. Fixture planning. In *Second Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.

[21] M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5):524–31, October 1988.

[22] A. Rao and K. Goldberg. Manipulating algebraic parts in the plane. *IEEE Transactions on Robotics and Automation*, 11(4), August 1995.

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

[24] U. Roy, C. Liu, and T. Woo. Review of dimensioning and tolerancing: Representation and processing. *Computer-Aided Design*, 23(7), 1991.

[25] F. van der Stappen, C. Wentink, and M. Overmars. Computing immobilizing grasps of polygonal parts. *Intl. Journal of Robotics Research*, 19(5):467–79, 2000.

[26] H. Voelker. A current perspective on tolerancing and metrology. *Manufacturing Review*, 6(4), 1993.

[27] H. Voelker. The current state of affairs in dimensional tolerancing. *Integrated Manufacturing Systems*, 9(4), 1998.

[28] R. Walker and V. Srinivasan. Creation and evolution of the ASME Y14.5M standard. *Manufacturing Review*, 7(1), 1994.

[29] Y. Wang. An optimum design approach to 3d fixture synthesis in a point set domain. *IEEE Transactions on Robotics and Automation*, 16(6), December 2000.

[30] J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. *Assembly Automation*, 17(2):129–36, August 1997.

[31] R. Wilhelm and S. C. Lu. *Computer Methods for Tolerance Design.* World Scientific, 1992.

[32] C. Yap and E.-C. Chang. Geometric tolerancing: Theory, issues, and computation. In *Second Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.