A Complete Algorithm for Designing Passive Fences to Orient Parts*

Jeff Wiegley† and Ken Goldberg‡ and Mike Peshkin§ and Mike Brokowski¶

Peshkin and Sanderson[18] showed that parts can be aligned as they move on a conveyor belt against a passive sequence of fences. In this paper we describe the first complete algorithm to design such sequences for a given convex polygonal part. The algorithm is complete in the sense that it is guaranteed to find a design if one exists and to terminate with a negative report otherwise. Based on an exact breadth-first search of the design space, the algorithm is also guaranteed to find the design requiring the fewest fences. We describe the algorithm and compare results with those previously reported. We conjecture that a fence design exists to orient any convex polygonal part.

In automated assembly it is often necessary to bring randomly oriented parts into uniform alignment. Often this is done mechanically, with passive devices such as the vibratory bowl feeder. The design of such feeders is currently an artform based on trial and error. Thus it is often time-consuming and error-prone. We seek to develop algorithms for the systematic design of feeders. Such algorithms would rapidly analyze part geometry based on friction and kinematics to assist in designing appropriate mechanisms for feeding a stream of such parts

We consider a class of feeders similar to that of vibratory bowls. These feeders, first described by Peshkin and Sanderson[18], translate parts past a sequence of passive fences using a standard conveyor belt. See Figure 1 for a pictorial example of one implementation. We consider the class of parts that can be effectively modelled as polygonal extrusions. Parts are singulated and arrive at random orientations. They move along a series of fences, each attached at some angle to the side walls of the conveyor. Adding curved tails to each fence aligns part edges with the fence prior to contact with the next fence [3]. After passing through the gauntlet, the part's final orientation should be uniquely determined. Such feeders differ from vibratory bowls in that parts are not rejected by filters; in the parlance of feeder design, all parts are passively "converted" to a unique desired orientation upon emerging from contact with a fence.

Figure 1: Example of a conveyor based system for orienting and feeding polygonal parts.

A feeder design is specified by a sequence of m fence angles. The design space, S^m, is uncountable. An initial algorithm [18] sampled the design space at uniform ($10°$) angles and thus was not complete in that it may fail to find a design that required fence angles other than those sampled. Moreover, even when it finds a design, there is no guarantee that it will find the design with fewest fences.

In this paper we describe an exact algorithm that partitions the design space into equivalence classes based on part geometry. This algorithm is guaranteed to find the shortest possible design if one exists. For the parts tested in [18], the new algorithm finds shorter designs and finds a fence design for one part that caused the previous method to fail.

This algorithm is based on two previous results:

*This work was supported by the National Science Foundation under Award IRI-9123747 and NSF Young Investigator Award IRI-9457523, by a grant from the State of California's Office of Competitive Technology, and by Adept Technology, Inc.

†Institute for Robotics and Intelligent Systems, Computer Science Department, University of Southern California.

‡UC Berkeley. goldberg@ieor.berkeley.edu. Part of this research was performed while Goldberg was at the University of Southern California.

§Department of Mechanical Engineering, Northwestern University.

¶ibid

1. Goldberg [9] reported a complete part feeding algorithm based on push-grasps with a parallel-jaw gripper. Based on an exact partition of the space of push-grasp angles into equivalence classes initially described in [11], the algorithm is guaranteed to find a sequence of active push-grasp operations to orient any polygonal part. The algorithm requires that the output of each operation can be modelled with a piece-wise constant monotone step function. As a result the algorithm can also be used to find a sequence of pure pushing motions to orient the part. The fences and algorithm described in [18] do not have this property: due to frictional uncertainty as the part rolls off the belt, the output is a one-to-many mapping that contains "bands" of possible part orientations.

2. Browkowski, Peshkin, and Goldberg [3] introduced curved fence tips to eliminate the above problem. Specialized curves can be generated by solving an equation that depends on part geometry; the paper also describes a more conservative class of "universal" curves that scale with part diameter. After sliding along this curve, part edges passively align with the fence. This eliminates the "band" of uncertainty in part orientation. As a result the effect of each fence angle can be modelled with a piecewise constant monotone function. Curved fences are similar to push grasps with one important difference: due to the unique direction of movement of the conveyor belt, adjacent fence angles are restricted to pushing the part from a restricted set of angles. Thus not every push plan can be transformed into a fence design and we cannot directly apply the algorithm from [9].

The first result gave a complete algorithm for motion planning, the second result introduced a relevant class of fences but did not give an algorithm for generating a sequence of fences. In this paper we combine these results to develop a complete algorithm that finds the shortest sequence of curved fences to align a given polygonal part.

1 Problem Definition

The input to the algorithm is a list of n rational coordinates describing a convex polygonal part, translated so that the origin coincides with the part's center of mass.

The output is a list of m rational angles describing the shortest sequence of fence angles that is guaranteed to orient a stream of such parts or report that no fence design exists.

We assume:

- The stream of identical parts is singulated prior to entering the first fence. This can be accomplished via a series of conveyor belts at increasing speeds,
- Part motion is planar,
- Parts are rigid and inertial forces are negligible,
- contact between fences and parts is frictionless,
- contact between parts and conveyor surface has some finite Culomb friction.

2 Related Work

Other methods for reducing uncertainty in part position and orientation have been studied. Erdmann and Mason[7] describe a system for orienting parts without sensors using tray tilting actions. The sides of their tray act somewhat similar to our fences. Brost [4] showed how to eliminate bounded uncertainty in part orientation using a parallel-jaw gripper. Balorda[2] outlined a method for reducing the position and uncertainty of a part using a single push with 2 point contacts. Mottaez and Goldberg[17] gave a method for removing bounded uncertainty in the position of a polygonal part using a sequence of pushes in the plane.

For cases where the initial pose of the part is known, Akella and Mason[1] present a planner for moving the part to a new position and orientation by pushing. They proved that if there are no obstacles, any part can be pushed between any two poses. Lynch[12] presents a method for determining pushes that maintain contact between a flat pusher and one edge of the part under frictional constraints. Lynch and Mason[13] gave an algorithm to plan paths for pushing a part from some intial pose to a goal pose in the presence of obstacles using such pushes.

Although the current paper assumes quasi-static interactions of parts, other authors have presented methods to treat part dynamics. Gilmore and Streit[8] present a rule-based system for predicting the dynamic behavior of parts as they move across fences. Mirtich and Canny[16] introduce an efficient method for modelling dynamic part behavior based on impulse simulation, which might be used to predict the dynamic behavior of a given fence design.

Brost[5] presents geometric analytical methods for representing the three dimensional configuration space obstacles formed by two contacting polygons. This permits analysis of interactions between non-convex parts. Building on this approach, Caine[6] studied the problem of designing vibratory bowl tracks and used configuration space obstacles to illustrate what part configurations can emerge from a given track design.

3 An Example

As an example Figure 2 shows one part from [18].

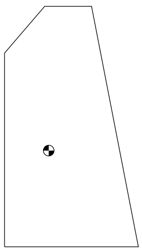


Figure 2: A part from the paper by Peshkin and Sanderson. Used by permission.
Vertices (ordered counterclockwise): $[(0,0), (1950,0), (1250,3250), (575,3250), (0,2625)]$,COM: (650,1300)

Figure 3 shows the fence design found by our algorithm. Computing this required 0.75 seconds on an Intel 486DX4-100 platform.

4 Fence Mechanics

As pointed out by [14], the mechanics of a polygonal part pushed under quasi-static conditions can be described using a function called the *radius function.* The radius of the part at angle θ is the distance along a line at angle θ passing through the part's center of mass measured from the COM to a line which just touches the boundary of the part and is perpendicular to θ. For a polygonal part, the radius function is piecewise sinusoidal with minima at stable orientations. See figure 4. When a part is pushed under quasi-static conditions it will rotate toward the nearest local minima as first proposed by Mason[14].

The *push* function for a part can be derived from the radius function. The push function maps initial orientations of the part to final orientations. See figure 5. The push function for a polygonal part is piecewise constant. The range of the function corresponds to local minima in the radius function, and discontinuities correspond to local maxima. Using the radius and push function it is possible to orient a part without sensors up to symmetry of the part's push function [9].

Let F be the push function for any part P. Then $F(\theta)$ is the final orientation achieved for a part being pushed at an angle θ. The push function F has the effect of partitioning the possible values for θ into equivalence classes. Visually we can recognize some of these equivalence classes as *steps* of the push function. Where for any step which begins at point α and ends at β, $F(\theta) = \phi$ for all $\theta \in [\alpha, \beta)$. Thus all pushes in such a range can be seen as achieving identical results. Additional equivalence classes arise from the concatenation of contiguous steps.

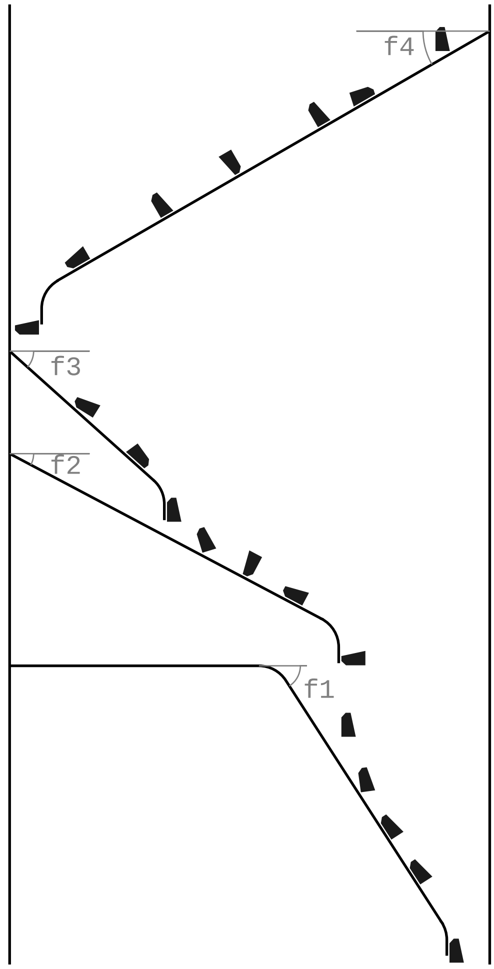


Figure 3: The fence design found by the complete algorithm. Fence Angles from top to bottom: $[f_4 = 30.0, f_3 = -41.9, f_2 = -27.9, f_1 = -57.2]$

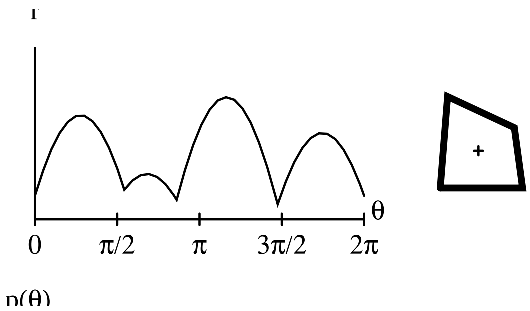


Figure 4: Radius function for the 4-gon shown at right.

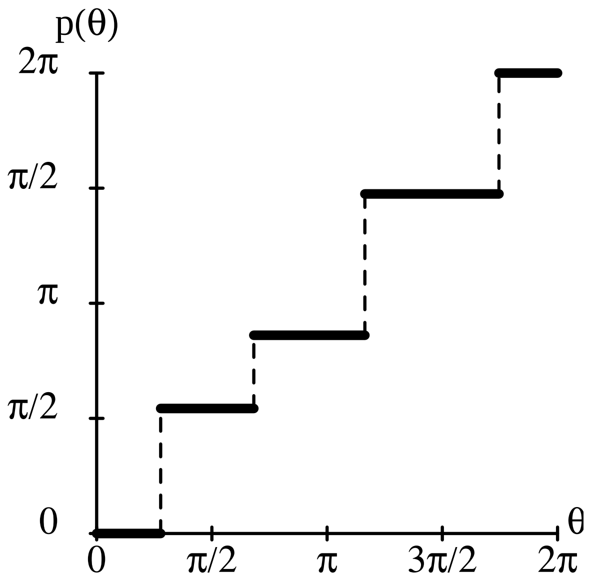


Figure 5: Push function for the part in the previous figure.

We represent such equivalence classes as push intervals or *p-intervals*. Where each p-interval is represented as a semi-closed interval of the form $[\alpha, \beta)$ such that α, β are points of discontinuity in the domain of push function F. For notation let α_i, β_i be the start and end points, respectively, for any p-interval p_i. Also let $|p_i|$ denote the Lebesgue measure of the p-interval p_i.

Let N be the number of stable edges for a part P then there are N such points of discontinuity in the domain of F. Let σ be the set of such points, then the set of equivalence classes, Σ, is $\{[\alpha, \beta)|\alpha, \beta \in \sigma\}$. Thus $|\Sigma| = N^2$

Now we can extend the definition of the function F to handle equivalence class inputs in an appropriate manner. Let $\delta = [\alpha, \beta)$ be some p-interval. Then define $F(\delta) = [\mu, \nu)$ where $\mu = F(\alpha), \nu = F(\beta)$. Note that the output $F(\delta)$ for any p-interval δ can itself be considered as a p-interval since its start and end points are also in the same domain as those of the p-intervals defined above (though not necessarily points of discontinuity).

5 The Algorithm

After the push function is used to partition the space of push directions into equivalence classes, we perform a breadth-first search of push combinations to find a fence design. To guide the search, we work backward from a unique final orientation toward a range of orientations of size 2π, which corresponds to the full range of uncertainty in initial part orientation. We then find fence angles in reverse order. In effect we find the last fence first and work upstream, like a salmon.

The search starts from a state which is a p-interval p_1 such that $F(p_1) = [\alpha, \alpha)$ for some α. Thus we start from a state for which we know of a single push angle θ_1 that results in a unique orientation for a known, limited range of starting orientations for the part. The search then proceeds from the current state, p_i to examine the next largest p-interval p_{i+1} such that $|F(p_{i+1})| <= |p_i|$. This implies then that any relative push angle θ such that $\alpha_{i+1} - \alpha_i < \theta < \beta_{i+1} - \beta_i$ will convert all orientations in p_{i+1} into some orientation in p_i. That is the desired relative push angle, θ_i, is equivalent to at least the differences of the starting points of the p-intervals p_{i+1} and p_i but no more than the difference of the end points of these p-intervals. We then continue searching with p_{i+1} as our current state. The goal of the breadth first search then is to arrive at a current state p_g such that $|p_g| = 2\pi$. Once a goal state is reached we have a sequence of desired relative push angles which we know will uniquely reorient a part regardless of its initial orientation because that initial orientation must be in the range of $[0, 2\pi) = p_g$.

This algorithm cannot be directly applied to the design of fences. Every fence must have a component of its push directed against the motion of the belt. This restricts combinations of relative push angles that can be realized with fences. The valid range for a fence depends on the angle of the previous fence (in particular whether it is attached to the left or right side of the conveyor). These constraints can be transformed into two contraints on relative push angles as illustrated in Figure 6.

$$\begin{array}{lcl} \theta_i \in Q_1 \cup Q_2 & \Longrightarrow & \theta_{i+1} \in Q_1 \cup Q_3 \\ \theta_i \in Q_3 \cup Q_4 & \Longrightarrow & \theta_{i+1} \in Q_2 \cup Q_4 \end{array}$$

The value for any fence angle f_i depends on the relative push angle θ_i, which is the *difference* of the p-intervals p_i and p_{i+1}, where the *difference* is defined as any angle in the range $(\alpha_{i+1} - \alpha_i, \beta_{i+1} - \beta_i)$. Any angle from this range is suitable for θ_i since all angles from this range achieves the same result. However, subranges of this range may not be suitable for a conveyor belt due to the constraints mentioned above. It is neccessary for the algorithm to examine each valid subrange. We initially pick the largest valid subrange, (α', β'), and search the other subranges breadth-first.

Given a valid range in a sequence it is neccessary to select a single value for θ_i for the push direction. For

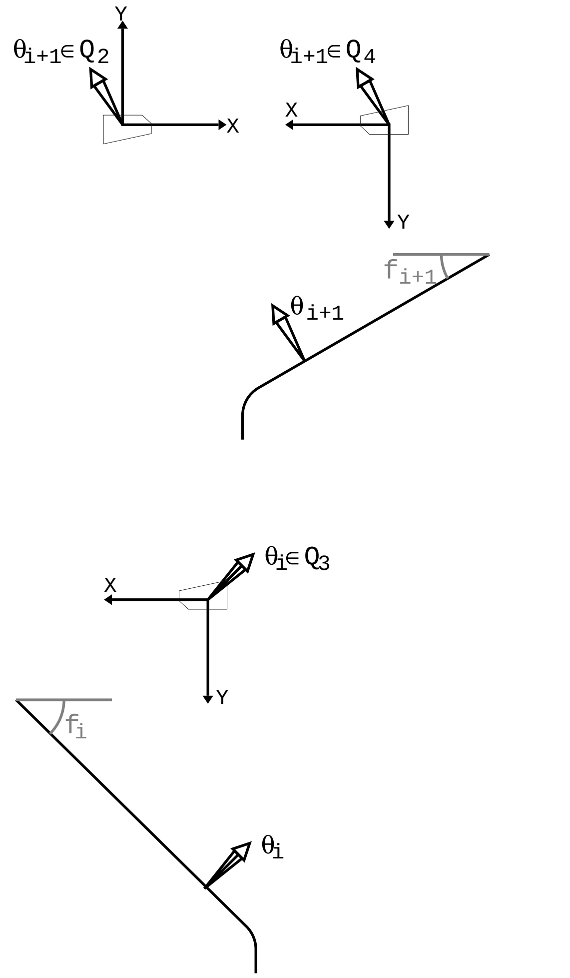


Figure 6: Conditions for $\theta_i \in Q_3 \implies \theta_{i+1} \in Q_2 \cup Q_4$. Similar figures justify the conditions for $\theta_i \in Q_1, Q_2, Q_4$.

our implementation we select $\theta_i = (\beta' - \alpha')/2$. The effect of this is that any part which has a rotational uncertainty less then $(\beta' - \alpha')/2$ (due to frictional effects or failure of certain assumptions for our model) will still be correctly oriented since the relative push actually occuring will be in the range (α', β'). In this manner the algorithm exhaustively searches all sequences of equivalence classes. If a plan is found it is guaranteed to be the shortest because of the nature of breadth first search and if the search fails to find any solution then no solution exists for the part. The search is guaranteed to halt since there are a finite number of equivalence classes and our search does not consider sequences with cycles.

Once the algorithm finds a sequence of relative push angles that obey the fence constraints, we must transform these to a sequence of fence angles, $f_1, \ldots, f_m$ that accomplishes the same push motions when implemented on a conveyor belt. The angles f_i can be determined by the function

$$f_i = f(\theta_i) = \begin{cases} \theta_i \in (0, \pi), & \theta_i - \pi/2 \\ \theta_i \in (\pi, 2\pi), & \theta_i + \pi/2 \end{cases}$$

5.1 Rational Angle Arithmetic

Floating-point arithmetic may alias two close but non-identical fence angles, thereby violating the completeness of the algorithm. We can avoid this problem using a method first suggested by Mason [15]. Since we assume that input part coordinates are rational, all transitions in the push function will occur at angles that can be represented exactly using coordinate pairs of arbitrary length integers that corresponde to the coordinates of part vertices. Every angle, θ, is internally represented by a pair of integers (x_θ, y_θ) such that $\theta = atan2(y_\theta, x_\theta)$. We refer to such angles as *rational.*

If we view the X-axis as representing the real component and the Y-axis as representing the imaginary component then every rational angle θ is described as a complex value $(\theta_r, \theta_i) = (x_\theta, y_\theta)$. Addition, subtraction and comparison of angles can then be done by simple multiplication and division of these complex values. For instance, consider two rational angles θ, ϕ, the addition $\theta + \phi$ is analogous to the multiplication.

$$\begin{aligned} \theta &= \theta_r + \theta_i i \\ \phi &= \phi_r + \phi_i i \\ \theta + \phi &= (\theta_r + \theta_i i)(\phi_r + \phi_i i) \\ &= \theta_r\phi_r + \theta_i\phi_i + (\theta_r\phi_i + \theta_i\phi_r)i \\ &= (\theta_r\phi_r + \theta_i\phi_i, \theta_r\phi_i + \theta_i\phi_r) \end{aligned}$$

Similarly, subtraction is analogous to division:

$$\begin{aligned} \theta - \phi &= \frac{\theta_r + \theta_i i}{\phi_r + \phi_i i} \\ &= \frac{(\phi_r + \phi_i i)(\theta_r + \theta_i i)}{\phi_r + \phi_i i} \\ &= \frac{\phi_r\theta_r - \phi_i\theta_i + (\phi_r\theta_i + \phi_i\theta_r)i}{(\phi_r + \phi_i i)^2} \\ &= \left(\frac{\phi_r\theta_r - \phi_i\theta_i}{(\phi_r + \phi_i i)^2}, \frac{\phi_r\theta_i + \phi_i\theta_r}{(\phi_r + \phi_i i)^2} \right) \\ &= (\theta_r\phi_r - \theta_i\phi_i, \theta_i\phi_i + \theta_i\phi_r) \end{aligned}$$

The last step in the above derivation can be used without loss of generality or accuracy since the magnitude of the real and imaginary portions does not affect the slope of the angle represented.

This allows us to entirely constrain the computation to the addition, subtraction and multiplication of arbitrary length integers. The result is that all additions and subtractions of rational angles are guaranteed to yield a rational angle. Comparison is done by simply comparing the slopes of the two rational angles.

We note that the set of angles in S^1 has cardinality $\aleph_1$. The cardinality of the set of representable rational angles is $|\{(x,y)|\text{integer}(x), \text{integer}(y)\}| = \aleph_0$. Thus there exist some angles which cannot be represented as rational angles. For instance the angle $\pi/3$.

5.2 Results

We implemented the algorithm based on a rational angle package in C++ that we based upon an earlier LISP implementation by Matt Mason. To test the design algorithm, we used a set of 31 random part shapes ranging from 2 to 7 stable sides. Finding an optimum fence design for these parts required an average of 1.25sec with only one part requiring more than 5 seconds (13.86sec) on an Intel 486DX4-100 platform.

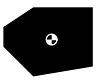
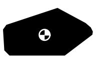
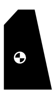
Part Shape	# Fences Using Algorithm [17]	# Fences Using New Algorithm
	3	2
	3	3
	8	5
	7	3
	X	3
	4	3
	8	3
	5	4

Figure 7: Results for a sample of parts.

Results for eight parts used as examples in the paper by Peshkin and Sanderson [18] are tabulated in Table 7. Note that for the fifth part in the table, the previous algorithm was unable to find a fence plan.

6 Discussion and Future Work

This paper describes the first complete algorithm for designing passive part feeders. This algorithm is *complete* in the sense that if a design exists the algorithm will find it else it will return with a negative report. Interestingly, even though fences must always push from a restricted set of directions, we have not been able to generate a single part for which a fence design does not exist. This leads us to conjecture that fences are solution-complete [10]: a design exists for all parts.

The next step is to extend these results to parts with curved edges and to relax the assumption of zero friction between parts and fences building on the results in [19].

References

[1] Srinivas Akella and Matthew T. Mason. An open-loop planner for posing polygonal objects in the plane by pushing. In *International Conference on Robotics and Automation*. IEEE, May 1992.

[2] Zdravko Balorda. Reducing uncertainty of objects by robot pushing. In *International Conference on Robotics and Automation*. IEEE, May 1990.

[3] Mike Brokowski, Michael A. Peshkin, and Ken Goldberg. Curved fences for part alignment. *ASME Journal of Mechanical Design*, 1992. (Accepted September 1993. A preliminary version of this paper appeared in the 1993 IEEE International Conference on Robotics and Automation, Atlanta, GA.).

[4] Randy C. Brost. Automatic grasp planning in the presence of uncertainty. *The International Journal of Robotics Research*, December 1988. Also appeared in Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April, 1986.

[5] Randy C. Brost. *Analysis and Planning of Planar Manipulation Tasks*. PhD thesis, CMU, January 1991.

[6] Michael E. Caine. *The Generation of Shape from Motion Constraint Specifications*. PhD thesis, MIT, 1993. To appear.

[7] Michael A. Erdmann and Matthew T. Mason. An exploration of sensorless manipulation. In *IEEE International Conference on Robotics and Automation*, 1986. Also appears in IEEE Journal of Robotics and Automation, V. 4.4, August 1988.

[8] B. J. Gilmore and D. A. Streit. A rule-based algorithm to predict the dynamic behavior of mechanical part orienting operations. In *International Conference on Robotics and Automation*. IEEE, 1988.

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

[10] Ken Goldberg. Completeness in robot motion planning. In *The Algorithmic Foundations of Robotics*. A. K. Peters, Boston, MA, 1995.

[11] Ken Goldberg and Matthew T. Mason. Bayesian grasping. In *International Conference on Robotics and Automation*. IEEE, May 1990.

[12] Kevin Lynch. The mechanics of fine manipulation by pushing. In *International Conference on Robotics and Automation*. IEEE, May 1992.

[13] Kevin Lynch and Matt Mason. Stable pushing: Mechanics, controllability, and planning. In *The First Workshop on the Algorithmic Foundations of Robotics*. A. K. Peters, Boston, MA, 1995.

[14] Matthew T. Mason. *Manipulator Grasping and Pushing Operations*. PhD thesis, MIT, June 1982. published in *Robot Hands and the Mechanics of Manipulation*, MIT Press, 1985.

[15] Matthew T. Mason. exact angle arithmetic package in commonlisp. (Ported to C++ by Jeff Wiegley (USC) in 1993)., August 1988.

[16] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In *The First Workshop on the Algorithmic Foundations of Robotics*. A. K. Peters, Boston, MA, 1995.

[17] Amir Mottaez, Murilo Coutinho, and Ken Goldberg. Positioning polygonal parts without sensors. In *Sensors and Controls for Automated Manufacturing Systems*. SPIE, September 1993.

[18] Michael A. Peshkin and Art C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5), October 1988.

[19] Anil Rao and Ken Goldberg. Friction and part curvature in parallel-jaw grasping. *Journal of Robotic Systems*, 12(6):365–382, June 1995.