Trap Design for Vibratory Bowl Feeders *

Robert-Paul Berretty [†‡] Ken Goldberg [§¶] Lawrence Cheung [§¶] Mark H. Overmars [†]
Gordon Smith [§¶] A. Frank van der Stappen [†]

Abstract

The vibratory bowl feeder is the oldest and still most common approach to the automated feeding (orienting) of industrial parts. In this paper we consider a class of vibratory bowl filters that can be described by removing polygonal sections from the track; we refer to this class of filters as *traps*. For an n-sided convex polygonal part and m-sided convex polygonal trap, we give an $O((n+m)\log(n+m))$ algorithm to decide if the part will be rejected by the trap, and an $O((nm(n+m))^{1+\epsilon})$ algorithm which deals with non-convex parts and traps. We then consider the problem of designing traps for a given part, and consider two rectilinear subclasses, *balconies* and *gaps* . We give linear and $O(n^2)$ algorithms for designing feeders and have tested the results with physical experiments using a commercial inline vibratory feeder. Our algorithms can be tested using online java applets: `http://ford.ieor.berkeley.edu/trap-design`.

1 Introduction

Part feeders, which singulate and orient parts prior to packing and insertion, are critical components of an automated assembly line. Although there is a substantial body of research in analytic feeder design, it has not yet produced a science base for practitioners, who still rely on instinct and rules-of-thumb [16]. Thus feeder design remains one of the biggest obstacles to automated manufacturing.

*Research is supported by NATO Collaborative Research Grant CRG 951224.

†Department of Computer Science, Utrecht University, PO Box 80089, 3508 TB Utrecht, The Netherlands.

‡Berretty's research is supported by the Dutch Organization for Scientific Research (N.W.O.)

§Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley, CA 94720, USA.

¶Goldberg and his students are also supported by the National Science Foundation under Presidential Faculty Fellow Award IRI-9553197 and CDA-9726389.

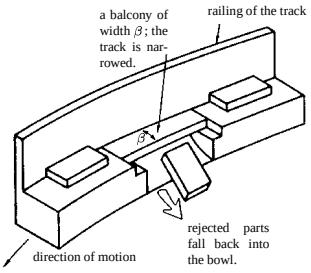

Figure 1: Vibratory bowl feeder track [7].

The oldest and still most common approach to automated feeding is the *vibratory bowl feeder* which consists of a bowl filled with parts surrounded by a helical metal track [7]. The bowl and track undergo an asymmetric helical vibration that causes parts to move up the track, where they encounter a sequence of mechanical devices such as wiper blades, grooves, gaps, and balconies. Most of these devices are filters that serve to reject (force back to the bottom of the bowl) parts in all orientations except for the desired one. Thus a stream of oriented parts emerges at the top after successfully running the gauntlet (See Figure 1).

In this paper we consider a class of vibratory bowl filters that can be described by removing polygonal sections from the track; we refer to this class of filters as *traps*. We first give algorithms to decide if a polygonal part will be rejected by the trap. We then consider the problem of designing traps for a given part, and consider two rectilinear subclasses, *gaps* and *balconies*. Proofs of some theorems can be found in a technical report [5].

2 Related Work

Space does not permit an adequate review of research in part feeding. An excellent introduction to mechanical parts feeders can be found in Boothroyd's book [7], which describes vibratory bowl feeders in detail as well as non-vibratory feeders such as the magnetic and revolving hook feeders. Sony introduced a novel approach using random

motion of parts over part-specific pallets [22, 20]. A variety of sensor-based alternatives to mechanical bowl feeders have been proposed. For example, Carlisle *et al.* [9] describe a system that combines machine vision with a high-speed robot arm and [18] use an optical silhouette sensor with air nozzle to reject parts on a feeder track.

Specific to vibratory bowls, researchers have used simulation [14, 4, 19], heuristics [16], and genetic algorithms [10] to design traps. Perhaps closest in spirit to our work is M. Caine's PhD thesis which develops geometric analysis tools to help designers by rendering the C-space for a given combination of part, trap, and obstacle [8].

Consider a part feeding system that accepts as input a set of part orientations Θ. Based on a definition by Akella *et al* [1], we might say that a system has the *feeding property* if there exists some orientation $\theta \in \Theta$ such that the system outputs parts only in orientation θ. This paper reports on algorithms that design traps with the feeding property. We are not aware of any previous algorithms for the systematic design of vibratory bowl traps.

3 Geometric Modeling

Throughout this paper, we focus on planar motion of the part as it slides across the track while maintaining contact with the vertical railing. In the figures, the railing is coincident with the x axis and the part moves in the positive x direction. We denote the part by P, its center of mass by c, and the trap by R. We focus on cases where P is a polygonal part with n vertices. The boundary of the trap belongs to the track, and therefore supports the part. The interior of the trap cannot support the part. We denote the interior of a shape by $\mathrm{int}(\cdot)$. The part of the track underneath P, supporting it is $S = P - \mathrm{int}(R)$.

The track is slightly tilted toward the railing so that the part remains in contact with the railing as it moves along the railing. The radius function for the part identifies stable orientations of the part against the railing [13].

Definition 3.1 *The* radius *of a part at an angle θ is the distance from the center of mass to the line tangent to the part, and orthogonally intersecting the ray from the center of mass in the direction of θ.*

Each stable orientation of P corresponds to a local minimum in the radius function. The stable orientations of the part can easily be computed in linear time from the description of the part [17]. There are only $O(n)$ stable orientations of the part against the railing.

We investigate the conditions that cause a part to drop through a trap. Let us for the sake of simplicity first fix position of the part with respect to the trap. Even when the part is partially supported, i.e. $S \neq \emptyset$, it might still drop.

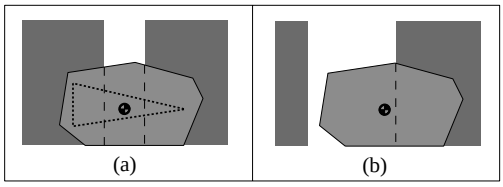

Figure 2: (a) A safe pose. The triangle is evidence of safeness. (b) An unsafe pose of the same part above a different trap.

Definition 3.2 *Let P be a part with center of mass c. Let R be a trap in the track of the bowl. The part P is* safe *if and only if there exists a triangle $\Delta = \{t_1, t_2, t_3\}$ with $c \in \Delta$, and $t_1, t_2, t_3 \in S$. Otherwise, P is* rejected.

In other words, there are three points of the part around the center of mass, which are supported by the track. Note, that the center of mass itself need not be supported.

We now state a lemma that gives a relation between the convex hull of the intersection of the part and the trap and the safeness of the part. Let $\mathcal{CH}(\cdot)$ denote the convex hull of a shape.

Lemma 3.3 *P is safe if and only if $c \in \mathcal{CH}(S)$.*

By a standard map overlay algorithm, $S = P - \mathrm{int}(R)$ can be computed in $O((n+m+k)\log(n+m))$ time. Here, k is the complexity of the intersection of the part and the trap. The convex hull can be computed in $O(k)$ time. The number of points, k, can be reduced to $O(n+m)$ by only taking into account the leftmost and rightmost intersection of the part and the trap, per edge of the trap. These intersections are computed using an $O((n+m)\log(n+m))$ sweepline algorithm. Hence, S is computed in $O((n+m)\log(n+m))$ time. (For these standard computational geometry algorithms we refer to the book of De Berg *et al.* [12]).

4 Analyzing a Trap

The analysis problem is to decide whether or not a part in a given orientation will be rejected (fall through) as it moves in the $+x$ direction along the railing and over the trap.

To analyze the safeness of the part, we consider the set of intersections between part and gap edges. This set changes as the part moves across the trap. We first determine which edges of the part are intersected by the edges of the gap as the part moves over the track. Second, we determine in which order the edges intersect each other. Therefore, we first sort the vertices of the part and the trap by their y-coordinate. Now, we have the y-coordinates of every edge, either from the trap, or the part.

We merge the edges of the part with the edges of the trap, and determine each x position of the part where a part edge

intersects with a trap edge. These "events" characterize the combinatorial changes in the intersection of the trap and the part during the motion. This preprocessing step takes $O(n+m)$ time in the convex case and $O(nm\log(nm))$ time otherwise.

A basic ingredient of our algorithm is to compute when the center of mass crosses a line defined by two intersecting pairs of trap and part edges. The position of an intersection point of a trap and a part edge is linearly dependent on the position of the part. This implies that the equation which describes the colinearity of the center of mass and the two intersection points is quadratic, and has at most two solutions.

The running time of the analysis algorithm is dependent on the shape of the part and the shape of the trap. If both the part and the trap are convex, then the problem is considerably easier to solve.

4.1 Convex Part, Convex Traps

For a convex part with a convex trap, the condition which describes the safeness of the part can be reformulated in terms of the vertices defined by intersections of the part and the trap edges.

Lemma 4.1 *Given a convex part P with center of mass c, and a convex trap R. P is safe if and only if c is in the convex hull of the vertices of $S \cap R$ or $c \notin R$.*

We restrict ourselves to the part of the motion when $c \in R$, which is a necessary condition for rejection of the part. We maintain $\mathcal{CH}(S \cap R)$, and check whether the center of mass is always inside $\mathcal{CH}(S \cap R)$.

As mentioned, during the motion of the part, intersections between the part and the trap edges may appear and disappear. Also, intersection points move. Fortunately, in our case, there are only four types of events at which the combinatorial structure of the convex hull changes. Details are in the technical report [5]. The four events occur when (1) a vertex of P moves across an edge of R, introducing or deleting a vertex of $\mathcal{CH}(S \cap R)$; (2) an edge of P moves across an vertex of R, introducing or deleting a vertex of $\mathcal{CH}(S \cap R)$; (3) a vertex of P moves across an edge of R, changing the defining edges of a vertex; (4) an edge of P moves across an vertex of R, changing the defining edges of a vertex.

Every event only requires a constant complexity update of the convex hull. By appropriately storing the convex hull, we can locate the place where the update is necessary in logarithmic time. Hence, the events can be handled in logarithmic time. After preprocessing, we know at which positions of the part, edges of the trap coincide with vertices of the part and vice versa. Therefore, maintaining the convex hull takes $O((n+m)\log(n+m))$ time.

During part motion, the center of mass always has the same distance to the edge of the track. Therefore, at any moment during the motion, there are only two edges of $\mathcal{CH}(S \cap R)$ that the center of mass can possibly cross. The intersecting edges of the trap and the part defining these edges might change, though. Every time the description of a relevant edge changes, a new event is generated for the position at which the center of mass (C.O.M.) will cross the new edge. This is accomplished without increasing the asymptotic running time. From the motion of the center of mass, and the motion of the relevant edges we derive poses of the part at which the center of mass leaves the convex hull. We add these poses as extra events. We handle such events as follows. We first check if the event is still valid, by checking if the edge associated with the event still is a relevant edge. If so, we report dropping of the part, otherwise, we discard the event. This gives no extra overhead for the algorithm. The following theorem summarizes the result.

Theorem 4.2 *Testing one orientation of a convex part against a convex trap can be accomplished in time $O((n+m)\log(n+m))$.*

4.2 Non-convex Part, Non-Convex Traps

In this section we briefly discuss how to test a pose of a not-necessarily-convex part against a not necessarily convex trap in the track. An approach with a promising time bound is the extension of the method of the previous section. Unfortunately, Lemma 4.1 is no longer valid. We must take into account supports of the part outside the trap. We have to maintain the convex hull of a set of nm moving and appearing and disappearing points. It turns out, that we can use an adaptation of the algorithm of Basch *et al.* [3, 2], leading to an algorithm with $O((nm(n+m))^{1+\epsilon})$ running time for any constant $\epsilon > 0$. We do not discuss this algorithm in this version of the paper, the interested reader is referred to the full version [5].

Theorem 4.3 *Testing one orientation of a polygonal part against a polygonal trap can be accomplished in $O((nm(n+m))^{1+\epsilon})$ time.*

5 Designing Traps

In this section we discuss how to design one trap such that the track satisfies the feeder property, i.e. only one orientation of the part survives the trap.

In the first subsection, we will discuss "gaps" as illustrated in Figure 3. In the second subsection we will discuss "balconies". The edges of gaps and balconies are orthogonal to the railing.

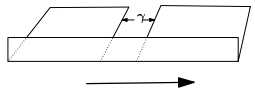

Figure 3: A gap in the track. The edges of the gap are orthogonal to the railing

5.1 The Gap

We assume, for a start, that P is convex. To get some flavor of the problem, we first show what happens if we move the part along a gap of arbitrary width. We start with the part to the left of the gap; the part is safe. During the motion, there might be a rejected position, if the gap is wide enough. At the end, when the center of mass is to the right of the right edge of the gap, the part is safe again. The position of the part at which the safeness of the part changes we call a *critical pose* (See Figure 5). If the gap is small enough, then the part remains safe throughout the whole motion, and there are no critical poses.

We focus on the *critical* gap-width, γ: the part passes safely over this gap but is rejected for gap-widths $\gamma + \epsilon$, for any $\epsilon > 0$. We compute the critical gap-width for a part in a given orientation along the railing.

The part is safe if and only if there is a supported triangle around the center of mass. This implies that if the part is rejected, then the supported area of the part is contained in a half-plane that does not contain the center of mass. We distinguish two different types of rejected poses of the part: (1) the part is only supported to the left (or the right) of the center of mass; (2) the supports are contained in a half-plane below (or above) the center of mass.

In the first type of rejected poses, the part can only be supported by one side of the gap, either the left or the right side. The second type of rejected poses correspond to poses in which the part is supported by both sides of the gap. See the last two frames of Figure 4.

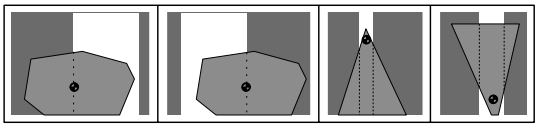

Figure 4: The types of rejected poses.

The critical gap-widths related to the first type of poses are relatively easy to compute by considering the radius function of the part at $\sigma - \frac{\pi}{2}$ and $\sigma + \frac{\pi}{2}$. Clearly, if one of these radii is less than the gap-width, then the part will fall either forward or backward. Thus, the critical gap-width γ is at most $\min\{\text{radius}(\sigma - \frac{\pi}{2}), \text{radius}(\sigma + \frac{\pi}{2})\}$.

The critical gap-width for the second type of critical poses is a bit harder to compute. Let us investigate how the supports of the part can be contained in a half-plane below the center of mass (the case for the supports above the center of mass is similar). The line defining the half-plane plays a crucial role in the analysis. Let us picture a part that is supported by two sides of the gap. The supported area of the part now consists of two convex regions, one to the left of the gap, and one to the right of the gap. The center of mass is contained in the gap. A half-plane extending downward and containing the supported area will always contain the entire lower half of the convex hull of the part. Therefore, the center of mass has to lie in the upper hull of the part to obtain an rejected pose.

A half-plane corresponding to a critical pose of the part is tangent to both the supported regions, as well as to the center of mass. In other words, the intersection points of the upper hull of the part and the gap, and the center of mass are colinear. Figure 5 shows an example of a critical pose of a convex part.

Computing the critical gap-width is accomplished by rotating a line around the center of mass, hereby sweeping over all critical poses of the part which in general have different gap-widths. The gap-widths corresponding to a critical pose is the horizontal distance between the intersection points of the line and the edges of the (upper hull of the) part. During the sweep, a linear number of pairs of edges are intersected by the line. For each such pair of edges of the upper hull of the part we compute the smallest gap-width such that there is a critical pose during the motion. The maximum of these gap-widths is the critical gap-width corresponding to the second type of rejected poses of the gap. This computation takes linear time.

Theorem 5.1 *For any orientation of the part, the critical gap-width can be computed in $O(n)$ time.*

Corollary 5.2 *Let P be a convex, polygonal part with n edges. In $O(n^2)$ time we can design a feeder with a gap, if such feeder exists, or report failure otherwise.*

If we drop the assumption that P is convex, the analysis is a bit more complex. But, it turns out that the critical gap-width can be computed in $O(n \log n)$ time, leading to an $O(n^2 \log n)$ algorithm to compute the feeder gap-width.

5.2 The Balcony

In this section, we treat a trap called a *balcony* which narrows the supporting surface of the track. Like a gap, a balcony is rectilinear. We can define this trap by giving the width, β, of the track. The trap is at least a long as the length of the part. See Figure 1 for an example.

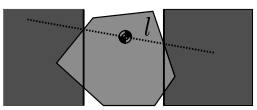

Figure 5: A critical pose.

For a balcony, this is also a sufficient condidtion.

Theorem 5.3 *Let P be a part with center of mass c. Let R be a balcony trap having width β. If at some moment during the motion, $c \in \text{int}(R)$, then the part will be rejected.*

Recall that the distance from c to the railing is exactly the radius of the part in the orientation it is traveling in. Hence, the theorem tells us that orientations with radius greater than the balcony-width drop off the track, and orientations with radius smaller than the balcony-width remain stable. The critical balcony-width for a given orientation is exactly its radius. Therefore, using a balcony, the orientation with the smallest radius can be selected by the bowl feeder. Clearly, this minimum can be computed in linear time.

Theorem 5.4 *Let P be a polygonal part with n vertices. In $O(n)$ time we can design a feeder using a balcony, if such feeder exists, or report failure otherwise.*

Note that the railing of the track always touches the part at the convex hull. Therefore, the given analysis holds for both convex and non-convex parts. A balcony can select orientations with a unique smallest radius. The only parts we cannot feed using a balcony are parts for which the minimal radius is not unique.

5.3 Parameterized Traps

In the previous two sections, we discussed simple traps such as gaps and balconies that can be described with only one parameter. More complicated traps can be specified with more parameters.

One way to look at the problem of designing a trap which is specified by k parameters, is to consider it as an arrangement of algebraic surfaces which divides a higher dimensional space into cells for which the part is safe, and cells for which the part is rejected. The space is spanned by the parameters of the trap, and the position of the part above the trap. The algebraic surfaces are derived from the higher dimensional boundaries of the convex hull of the part and the trap in different configurations. This is very much in the flavor of general robot motion planning using a cell decomposition approach. We refer the reader to Latombe's book [15] for an overview of robot motion planning, and to the paper of Schwartz and Sharir [21] for a solution to the general motion planning problem. Computing and processing the cells can be done by Collins' cylindrical algebraic decomposition [11], and is therefore doubly exponential in k. Details and possible improvements are in [5].

6 Experimental Results

To facilitate study, we implemented a java applet that allows users on the internet to experiment with our trap analysis and design algorithms. With a mouse, users can create a polygonal part. The applet automatically designs traps to feed the given part and demonstrates the results with animation: `http://ford.ieor.berkeley.edu/trap-design`.

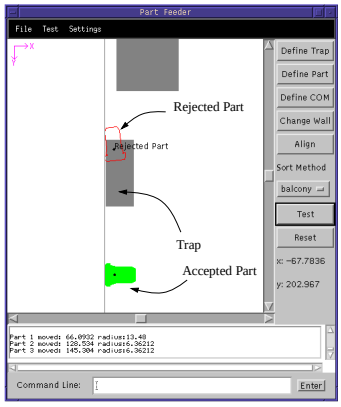

Figure 6: Trap Design Applet.

One function in the applet is an analytic test to decide if a polygonal part, in its current position and orientation, will fall through a given trap. For a polygonal part sliding across a trap, the applet predicts when, if ever, the part reaches a point where it will fall in.

Instead of using the algorithm of Section 4.2, we currently use a simple algorithm which does a repeated search for support triangles. Even though the theoretical running time of this algorithm is rather poor ($O((nm)^5)$), it runs fast in practice.

The applet is also able to design balconies using the algorithm of Section 5.2. It computes the radius function of the user's part. If there is a unique minimum value, a balcony with this width will ensure that all other poses with a larger radius will fall prey to the trap.

The applet can also design a type of trap we did not discuss in the paper. This trap consists of a "bridge" or pair of inner and outer balconies. Two parameters define this trap: the width of the bridge and width of the balcony at the side of the railing.

After designing the traps, the applet simulates the behavior of the parts as they are randomly rotated and fall against the railing, and then move along the track. We are currently working to incorporate gap design into the applet.

Also, we built a physical inline vibratory feeder and succesfully tested various traps. See Figure 7 for a picture of our feeder.

7 Conclusion

In this paper, we report algorithms for the planar analysis and design of traps for polygonal parts moving across a

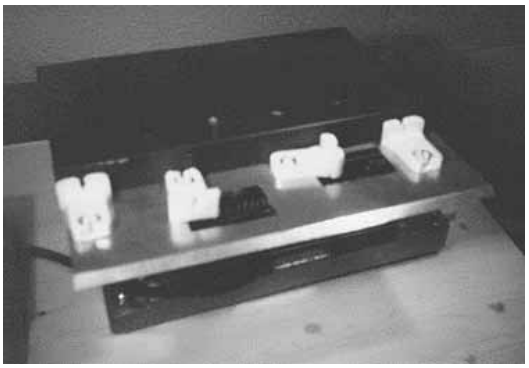

Figure 7: Part on track and railing mounted on Model 5300A.1 (T-18) adjustable inline vibratory feeder from Automation Devices, Inc. Approximate length 18 inches. The traps were designed by the algorithm and cut with a milling machine. The feeders successfully feeds a stream of these parts.

feeder track. We are not aware of any previous algorithms for the systematic design of vibratory bowl traps.

Treating parts and motion in all of their three-dimensional glory will require additional work, for example consider the common technique used for feeding screws, where a slot is used such that the screw shaft drops in, but the screw head prevents the screw from falling back into the bowl. We are working to extend our analysis to 3D.

We are also investigating systematic algorithms for the design of other commonly used feeder devices such as blades, wipers, and ramps. Armed with these tools, we can then address the planning problem, which is to automatically design an optimal sequence of such devices to deliver a desired part orientation.

Our goal is to develop complete algorithms for synthesizing feeder tracks: given part geometry, a complete algorithm will either find a track if one exists or report that no track exists for this part. Complete algorithms have been reported for feeding with parallel-jaw grippers [13] and for fences on a conveyor belt [6].

References

[1] S. Akella, W. Huang, K. Lynch, and M. Mason. Sensorless parts feeding with a one joint robot. *Algorithms for Robotic Motion and Manipulation*, 1996.

[2] J. Basch. Private communication. 1998.

[3] J. Basch, Leonidas J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[4] D. Berkowitz and J. Canny. Designing parts feeders using dynamic simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, 1996.

[5] R.-P. Berretty, K. Goldberg, L. Cheung, M.H. Overmars, G. Smith, and A.F. van der Stappen. Trap design for vibratory bowl feeders. Technical report, Department of Computer Science, Utrecht University, 1999. To appear.

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

[7] G. Boothroyd, C. Poli, and L. Murch. *Automatic Assembly*. Marcel Dekker, Inc., New York, 1982.

[8] Mike Caine. The design of shape interactions using motion constraints. In *IEEE International Conference on Robotics and Automation*, 1994.

[9] Brian Carlisle, Ken Goldberg, Anil Rao, and Jeff Wiegley. A pivoting gripper for feeding industrial parts. In *International Conference on Robotics and Automation*. IEEE, May 1994.

[10] A. Christiansen, A. Edwards, and C. Coello. Automated design of parts feeders using a genetic algorithm. In *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, 1996.

[11] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes Comput. Sci.*, pages 134–183. Springer-Verlag, Berlin, West Germany, 1975.

[12] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[13] Ken Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2):201–225, August 1993.

[14] M. Jakiela and J. Krishnasamy. Computer simulation of vibratory parts feeding and assembly. In *2nd International Conference on Discrete Element Methods*, March 1993.

[15] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[16] L. Lim, B. Ngoi, S. Lee, S. Lye, and P. Tan. A computer-aided framework for the selection and sequencing of orientating devices for the vibratory bowl feeder. *International Journal of Production Research*, 32(11), 1994.

[17] M. Mason. *Manipulator grasping and pushing operations*. PhD thesis, MIT, 1982. published in *Robot Hands and the Mechanics of Manipulation*, MIT Press, Cambridge, 1985.

[18] G. Maul and N. Jaksic. Sensor-based solution to contiguous and overlapping parts in vibratory bowl feeders. *Journal of Manufacturing Systems*, 13(3), 1994.

[19] G. Maul and M. Thomas. A systems model and simulation of the vibratory bowl feeder. *Journal of Manufacturing Systems*, 16(5), 1997.

[20] Paul Moncevicz, Mark Jakiela, and Karl Ulrich. Orientation and insertion of randomly presented parts using vibratory agitation. In *ASME 3rd Conference on Flexible Assembly Systems*, September 1991.

[21] J. T. Schwartz and Micha Sharir. On the "piano movers" problem II: general techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4:298–351, 1983.

[22] Sony. Advanced parts orienting system (apos). Technical Report 801-8902-11, Sony Corporation, 1989.