Geometry and Part Feeding

A. Frank van der Stappen [1], Robert-Paul Berretty [1,2],
Ken Goldberg [3], and Mark H. Overmars [1]

[1] Institute of Information and Computing Sciences,
Utrecht University,
P.O.Box 80089, 3508 TB Utrecht, the Netherlands

[2] Current address: Department of Computer Science,
University of North Carolina,
Campus Box 3175, Sitterson Hall, Chapel Hill, NC 27599-3175, USA

[3] Department of Industrial Engineering and Operations Research,
University of California at Berkeley,
Berkeley, CA 94720, USA

Abstract. Many automated manufacturing processes require parts to be oriented prior to assembly. A part feeder takes in a stream of identical parts in arbitrary orientations and outputs them in uniform orientation. We consider part feeders that do not use sensing information to accomplish the task of orienting a part; these feeders include vibratory bowls, parallel jaw grippers, and conveyor belts and tilted plates with so-called fences. The input of the problem of sensorless manipulation is a description of the part shape and the output is a sequence of actions that moves the part from its unknown initial pose into a unique final pose. For each part feeder we consider, we determine classes of orientable parts, give algorithms for synthesizing sequences of actions, and derive upper bounds on the length of these sequences.

1 Introduction

Manipulation tasks such as part feeding generally take place in structured factory environments; parts typically arrive at a more-or-less regular rate along for example a conveyer belt. The structure of the environment removes the need for intricate sensing capabilities. In fact, Canny and Goldberg [22] advocate a RISC (Reduced Intricacy in Sensing and Control) approach to designing manipulation systems for factory environments. Inspired by Whitney's recommendation that industrial robots have simple sensors and actuators [38], they argue that automated planning may be more practical for robot systems with fewer degrees of freedom (parallel-jaw grippers instead of multi-fingered hands) and simple, fast sensors (light beams rather than cameras). To be cost-effective industrial robots should emphasize efficiency and reliability over the potential flexibility of anthropomorphic designs. In addition to these advantages of RISC hardware, RISC systems also lead to positive effects in software: manipulation algorithms that are efficient, robust, and subject to guarantees.

We consider part feeders in the line of thought of the RISC approach. More specifically, we shall focus on the problem of sensorless orientation of parts in which *no* sensory information at all is used to move the part from an unknown initial pose into a unique—and known—final pose. In sensorless orientation or part feeding parts are oriented using passive mechanical compliance. The input of the problem of sensorless orientation is a description of the shape of the part and the ouput is a sequence of open-loop actions that moves the part from an unknown initial pose into a unique final pose. Among the sensorless part feeders considered in the literature are the traditional bowl feeder [18, 19], the parallel-jaw gripper [23, 26], the single pushing jaw [3, 29, 31, 34], the conveyor belt with a sequence of fences rigidly attached to both its sides [20, 35, 39], the conveyor belt with a single rotational fence [2], the tilting tray [25, 33], and vibratory plates and programmable vector fields [16, 17].

Traditionally, sensorless part feeding is accomplished by the *vibratory bowl feeder*, which is a bowl that is surrounded by a helical metal track and filled with parts [18, 19], see Figure 1. The bowl and track undergo an asymmetric

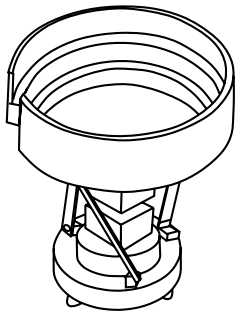


Fig. 1. A bowl feeder [19].

helical vibration that causes parts to move up the track, where they encounter a sequence of mechanical devices such as wiper blades, grooves and traps. The majority of these mechanical devices act as filters that serve to reject (force back to the bottom of the bowl) parts in all orientations except for the desired one. Eventually, a stream of parts in a uniform orientation emerges at the top after successfully running the gauntlet. The design of bowl feeders is, in practice, a task of trial and error. It typically takes one month to design a bowl feeder for a specific part [30]. We will see in Section 5 that it is possible to compute whether a given part in a given orientation will safely move across a given trap. More importantly, we will see that it is possible to use the knowledge of the shape of the part to synthesize traps that allow the part to pass in only one orientation [9, 12, 13].

The first feeders to which thorough theoretical studies have been devoted were the parallel-jaw gripper and pushing jaw. Goldberg [26] showed that these

devices can be used for sensorless part feeding or orienting of two-dimensional parts. He gave an algorithm for finding the shortest sequence of pushing or squeezing actions that will move the part from an unkown initial orientation to a known final orientation. Chen and Ierardi [23] showed that the length of this sequence is $O(n)$ for polygonal parts with n vertices. In Section 2 we shall provide theoretical foundation to the fact that the sequence length often stays well below this bound [37]. As the act of pushing is common to most feeders that we consider in this paper we will first study the pushing of parts in some detail.

The next feeder we consider consists of a sequence of fences which are mounted across a conveyor belt [20, 35, 39]. The fences brush the part as it travels down the belt thus reorienting it (see Figure 2). The motion of the belt effectively

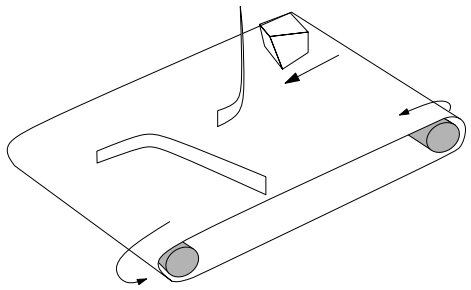


Fig. 2. Rigid fences over a conveyor.

turns the slide along a fence into a push action by the fence. It has long been open whether a sequence of fences can be designed for any given part such that this sequence will move that part from any initial pose into a known final pose. We report an affirmative answer in Section 3. In addition we give an $O(n^3)$ algorithm (improving an earlier exponential algorithm by Wiegley et al. [39]) for computing the shortest sequence of fences for a given part along with several extensions [8, 10, 11].

A drawback of most of the achievements in the field of sensorless orientation of parts is that they only apply to planar parts, or to parts that are known to rest on a certain face. In Section 4 we present a generalization of conveyor belts and fences that attempts to bridge the gap to truly three-dimensional parts [15]. The feeder consists of a sequence tilted plates with curved tips; each of the plates contains a sequence of fences (see Figure 3). The feeder essentially tries to orient the part by a sequence of push actions by two orthogonal planes. We analyze these actions and use the results to show that it is possible to compute a set-up of plates and fences for any given asymmetric polyhedral part such that the part gets oriented on its descent along plates and fences.

This paper reports on parts of our research in the field of sensorless manipulation of the last few years. The emphasis will be on the transformation of various sensorless part feeder problems into geometric problems, a sketch of the

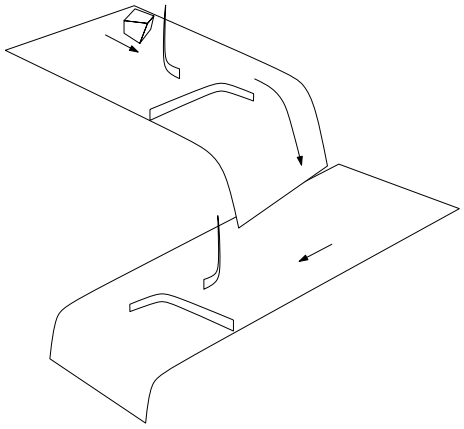


Fig. 3. Feeding three-dimensional parts with a sequence of plates and fences.

algorithms that solve these problems, and on determining classes of orientable parts. For proofs and detailed descriptions of the algorithms and their extensions the reader is in general referred to other sources [8–15, 37].

2 Pushing planar parts

2.1 Push functions

Throughout the entire paper, we assume zero friction—unless stated otherwise—between the part and the orienting device. Let c be the center-of-mass and P be the convex hull of the planar part. As a pushing device always touches the part at its convex hull, we can only orient a part up to rotational symmetries in its convex hull. Without loss of generality, our problem is now to orient the convex part P with given center-of-mass c.

We assume that a fixed coordinate frame is attached to P. Directions are expressed relative to this frame. The *contact direction* of a supporting line (or tangent) l of a part P is uniquely defined as the direction of the vector perpendicular to l and pointing into P (see Figure 4 for a supporting line with contact direction π.). As in Mason [31], we define the *radius function* $\rho : [0, 2\pi) \rightarrow \{x \in \mathbb{R} | x \geqslant 0\}$ of a part P with a center-of-mass c; ρ maps a direction ϕ onto the distance from the center-of-mass c to the supporting line of P with contact direction ϕ. Recall that the direction ϕ is measured with respect to the frame attached to P. The (continuous) radius functions determines the push function, which, in turn, determines the final orientation of a part that is being pushed.

Throughout this paper, parts are assumed to be pushed in a direction perpendicular to the pushing device. The *push direction* of a single jaw is determined by the direction of its motion. The push direction of a jaw pushing a part equals the contact direction of the jaw. In most cases, parts will start to rotate when

pushed. If pushing in a certain direction does *not* cause the part to rotate, then we refer to the corresponding direction as an *equilibrium* (push) *direction* or *orientation*. These equilibrium orientations play a key role throughout this paper. If pushing does change the orientation, then this rotation changes the orientation of the pushing gripper relative to the part. We assume that pushing continues until the part stops rotating and settles in a (stable) equilibrium pose.

The *push function* $p : [0, 2\pi) \to [0, 2\pi)$ links every orientation ϕ to the orientation $p(\phi)$ in which the part P settles after being pushed by a jaw with push direction ϕ (relative to the frame attached to P). The rotation of the part due to pushing causes the contact direction of the jaw to change. The final orientation $p(\phi)$ of the part is the contact direction of the jaw after the part has settled. The equilibrium push directions are the fixed points of the push function p.

The push function p consists of *steps*, which are intervals $I \subset [0, 2\pi)$ for which $p(\phi) = v$ for all $\phi \in I$ and some constant $v \in I$, and *ramps*, which are intervals $I \subset [0, 2\pi)$ for which $p(\phi) = \phi$ for all $\phi \in I$. Note that the ramps are intervals of equilibrium orientations. The steps and ramps of the push function are easily constructed [26, 36] from the radius function ρ, using its points of horizontal tangency; these orientations of horizontal tangency are the equilibrium push orientations. Angular intervals of constant radius turn up as ramps of the push function. Notice that such intervals only exist if the boundary of the part contains certain specific circular arcs. Thus, ramps cannot occur in the case of polygonal parts. If the part is pushed in a direction corresponding to a point of non-horizontal tangency of the radius function then the part will rotate in the direction in which the radius decreases. The part finally settles in an orientation corresponding to a local minimum of the radius function. As a result, all points in the open interval I bounded by two consecutive local maxima of the radius function ρ map onto the orientation $\phi \in I$ corresponding to the unique local minimum of ρ on I. (Note that ϕ itself maps onto ϕ because it is a point of horizontal tangency.) This results in the steps of the push function. Note that each half-step, i.e., a part of a step on a single side of the diagonal $p(\phi) = \phi$, is a (maximal) angular interval without equilibrium push orientation. An equilibrium orientation v is *stable* if it lies in the interior of an interval I for which $p(\phi) = v$ for all $\phi \in I$. Besides the steps and ramps there are isolated points satisfying $p(\phi) = \phi$ in the push function, corresponding to local maxima of the radius function. Figure 4 shows an example of a radius function and the corresponding push function.

Similar to the push function we can define a *squeeze function* that links every orientation ϕ to the orientation in which the part settles after being simultaneously pushed from the directions ϕ and $\phi + \pi$. The steps and ramps of the squeeze function can be computed from the part's *width function* (see [26, 36] for details).

Using the abbreviation $p_\alpha(\phi) = p((\phi + \alpha) \bmod 2\pi)$, we define a *push plan* to be a sequence $\alpha_1, \ldots, \alpha_k$ such that $p_{\alpha_k} \circ \ldots \circ p_{\alpha_1}(\phi) = \Phi$ for all $\phi \in [0, 2\pi)$ and a fixed Φ. In words, a push plan is an alternating sequence of jaw reorienations—by angles α_i—and push actions that will move the part from any initial orientation ϕ into the unique final orientation Φ. Observe that a single push action puts

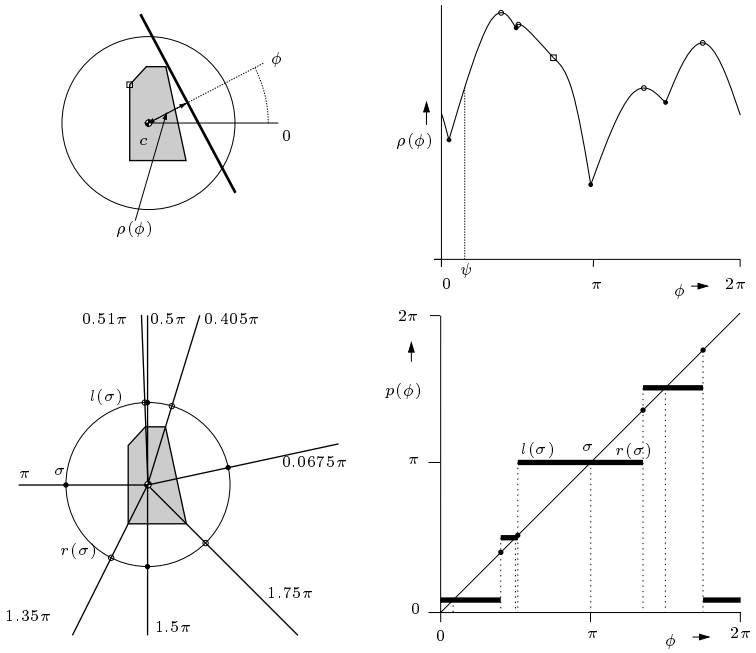


Fig. 4. A polygonal part and its radius and push function. The minima of the radius function correspond to normals to polygon edges that intersect the center-of-mass. The maxima correspond to tangents to polygon vertices whose normals intersect the center-of-mass. The horizontal steps of the push function are angular intervals between two successive maxima of the radius function.

the part into one of a finite number of stable orientations. Most algorithms for computing push plans proceed by identifying reorientations that will cause a next push to reduce the number of possible orientations of the part.

2.2 Push plan length

Goldberg [26] considered the problem of orienting (feeding) polygonal parts using a parallel-jaw gripper. A parallel-jaw gripper consists of a pair of flat parallel jaws that can close in the direction orthogonal to the jaws, which can *push* and *squeeze* the part. When the initial orientation of the part is unknown, a sequence of gripper operations can be used to orient the part—relative to the gripper—without sensing. Let N be the number of gripper operations in the shortest sequence that will orient the part up to symmetry. Goldberg showed that N is $O(n^2)$ for polygonal parts with n vertices and gave an algorithm for finding the shortest squeeze plan. He also conjectured that N is $O(n)$.

Chen and Ierardi [23] proved Goldberg's conjecture by constructing simple push and squeeze plans of length $O(n)$. They also presented pathological polygons where N is $\Theta(n)$, showing that the $O(n)$ bound is tight in the worst case.

Such pathological polygons are 'fat' (approximately circular), while N is almost always small for 'thin' parts. Consider the two parts shown in Figure 5. Imagine

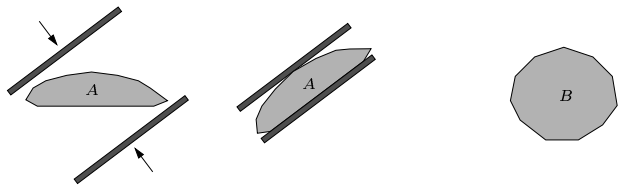


Fig. 5. Both polygonal parts have $n = 11$ vertices, but part A is thin, while B is fat. Part A is intuitively easier to orient than part B.

grasping part A. Regardless of the orientation of the gripper, we expect the part to be squeezed into an orientation in which its longest edge is aligned with a jaw of the gripper. Hence, the number of possible orientations of the part (relative to the gripper) after a single application of the gripper is very small. Part B can end up with any of its n edges against a gripper jaw; the number of possible orientations (again relative to the gripper) after a single application of the gripper is considerably higher than in the case of the thin part. In general, we observe that thin parts are easier to orient than fat ones.

A theoretical analysis confirms this intuition. To formalize our intuition about fatness, we define the *geometric eccentricity* of a planar part based on the length-to-width ratio of a distinguished type of bounding box. We deduce an upper bound on the number of actions required to orient a part that depends only on the eccentricity of the part. The bound shows that a constant number of actions suffices to orient a large class of parts. The analysis also applies to curved parts and provides the first complexity bound for non-polygonal parts.

The inspiration for our thinness measure comes from ellipses. The eccentricity of an ellipse equals $\sqrt{1-(b/a)^2}$, where a and b are the lengths of the major and minor axes respectively. Our (similar) definition of eccentricity for a convex object relies on the maximum of all aspect ratios of bounding boxes of the object.

Definition 1 *The* eccentricity ϵ *of a convex object* $P \subset \mathbb{R}^2$ *is defined by* $\epsilon = r - 1$, *where* r *equals the maximum of all aspect ratios of bounding boxes of* P.

Note that the minimum eccentricity of 0 is in both our definition and in the definition for ellipses obtained for circles.

Chen and Ierardi [23] proposed a class of plans for orienting polygonal parts based on repeating a unique push-and-reorient operation. The length of the longest angular interval without equilibrium orientation, or, in other words, of the longest half-step of the push function, determines the angle of reorientation. Assume that this half-step is uniquely defined and has length α. A reorientation by $\alpha - \mu$ for some very small positive μ in the proper direction followed by a push action will cause the part to rotate to the next equlibrium orientation unless it

is in the orientation ϕ corresponding to the height—in the push function—of the longest half-step. Since the number of steps is bounded by n, it will take at most n of these combined actions to make the part end up in orientation ϕ. The case where the longest half-step is not uniquely defined requires additional techniques but again a plan of linear length can be obtained.

We use ideas similar to those of Chen and Ierardi to establish a relation between the length of the longest half-step and the number of push actions required to orient the part. The bound applies to arbitrary parts and is given in the following lemma.

Lemma 1. *A part can be oriented by $N = 2\lceil 2\pi/\alpha \rceil + 1$ applications of the gripper, where α is the longest-half-step of the push function.*

Eccentricity imposes a lower bound on the length of the longest half-step. Intuitively it is clear that a part can only be eccentric when its radius is allowed to increase over a relatively long angular interval (about its center-of-mass). A thorough analysis [37] confirms this intuition. The result of the analyis is given below.

Lemma 2. *The eccentricity ϵ of a part with a push function with a longest half-step of length α is bounded by*

$$\epsilon \leqslant \frac{\cos^{k-1}\alpha \cdot \sin(k+1)\alpha}{\cos^{k} 2\alpha} - 1,$$

where $k = \lceil \pi/(2\alpha) \rceil$.

Lemmas 1 and 2 yield the following theorem.

Theorem 1. *Let P be a part with eccentricity*

$$\epsilon > \frac{\cos^{k-1}\alpha \cdot \sin(k+1)\alpha}{\cos^{k} 2\alpha} - 1$$

($k = \lceil \pi/(2\alpha) \rceil$), for some $\alpha \in (0, \pi/4)$. Then, P can be oriented by a push plan of length

$$N \leqslant 2\lceil \frac{2\pi}{\alpha} \rceil + 1.$$

Theorem 1 shows that the number of push actions needed to orient a part is a function of its eccentricity. It provides the first upper bound on the length of a push plan for non-polygonal parts. Sample values show that the upper bound provided by Theorem 1 is relatively low even for smaller values of ϵ; $N \approx 75$ for $\epsilon = 0.5$, N is below 50 for $\epsilon = 1$ and below 30 for $\epsilon = 2.5$. Similar bounds can be obtained for squeeze plans [37].

2.3 Pulling parts

We have recently studied sensorless orientation of planar parts with elevated edges by inside-out pull actions [14]. In a pull action a finger is moved (from the

inside of the part) towards the boundary. As it reaches the boundary it continues to pull in the same direction until the part is certain to have stopped rotating. Subsequently, the direction of motion of the finger is altered and the action is repeated. The problem os sensorless orientation by a pulling finger is to find a sequence of motion directions that will cause the finger to move the part from any initial pose into a unique final pose.

Although intuitively similar to pushing it turns out that sensorless orienting by pull actions is considerably harder than pushing [14]. As the finger touches the part from the inside it does no longer make sense to assume that the part is convex. In fact, it can be shown that certain non-convex parts cannot be oriented by a sequence of pull actions. Most convex parts are orientable by $O(n)$ pull actions, and the shortest pull plan is computable in $O(n^3)$ time.

3 Fence design

The problem of *fence design* is to determine a sequence of fence orientations, such that fences with these orientations align a part as it moves down a conveyor belt and slides along these fences [20, 35, 39]. Figure 6 shows a fence design that orients the given part regardless of its initial orientation. We shall see below that fence design can be regarded as finding a constrained sequence of push directions. The additional constraints make fence design considerably more difficult than sensorless orientation by a pushing jaw.

Wiegley *et al.* [39] gave an exponential algorithm for computing the shortest sequence of fences for a given part, if such a sequence exists. They conjectured that a fence design exists for any polygonal part. We prove the conjecture that a fence design exists for any polygonal part. In addition, we give an $O(n^3)$ algorithm for computing a fence design of minimal length (in terms of the number of fences used), and discuss extensions and possible improvements.

We address the problem of designing a shortest possible sequence of fences $f_1, \ldots, f_k$ that will orient P when it moves down a conveyor belt and slides along these fences. Let us assume that the conveyor belt moves vertically from top to bottom, as indicated in the overhead view in Figure 6. We distinguish between left fences, which are placed along the left belt side, and right fences, which are placed along the right side. The angle or orientation of a fence f_i denotes the angle between the upward pointing vector opposing the motion of the belt and the normal to the fence with a positive component in upward direction. The motion of the belt turns the sliding of the part along a fence into a push by the fence. The direction of the push is—by the zero friction assumption—orthogonal to the fence with a positive component in the direction opposing the motion of the belt. Thus, the motion of the belt causes any push direction to have a positive component in the direction opposing the belt motion. We now transform this constraint on the push direction relative to the belt into a constraint on successive push directions relative to the part.

Sliding along a fence f_i causes one of P's edges, say e, to align with the fence. The carefully designed [20] curved tip of the fence guarantees that e is

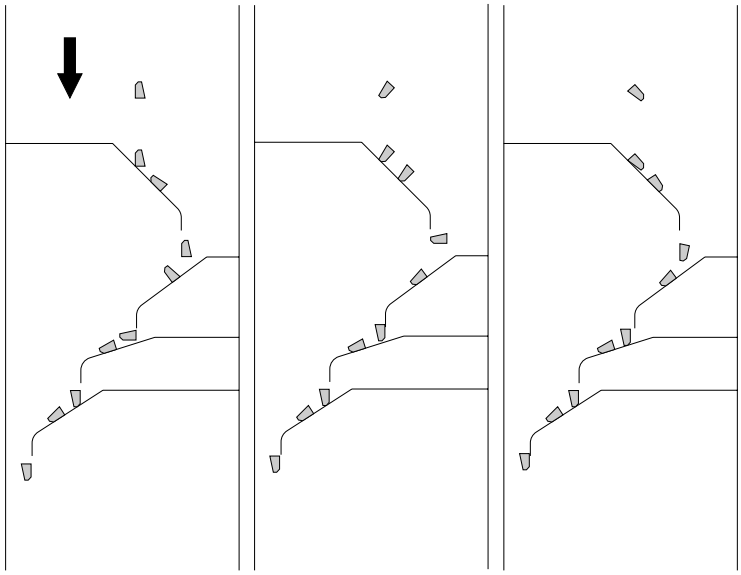


Fig. 6. Three overhead views of the same conveyor belt and fence design. The traversals for three different initial orientations of the same part are displayed. The traversals show that the part ends up in the same orientation in each of the three cases.

aligned with the belt sides as P leaves the fence. If f_i is a left (right) fence then e faces the left (right) belt side (see Figure 7). Assume f_i is a left fence. The reorientation of the push direction is the difference between the final contact direction of f_i and the initial contact direction of f_{i+1}. At the moment of leaving f_i, the contact direction of f_i is perpendicular to the belt direction and towards the right belt side. So, the reorientation of the push direction is expressed relative to this direction.

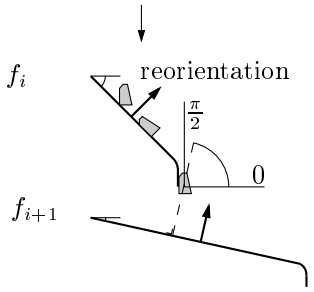
Figure 7(a) shows that the reorientation α_{i+1} is in the range $(0, \pi/2)$ if we choose f_{i+1} to be a left fence. If we take a right fence f_{i+1} then the reorientation is in the range $(\pi/2, \pi)$. A similar analysis can be done when P leaves a right fence and e faces the left belt side. The results are given in Figure 7(b).

The table shows that the type t_i of fence f_i imposes a bound on the reorientation α_{i+1}. Application of the same analysis to fences f_{i-1} and f_i and reorientation α_i leads to the following definition of a valid fence design [39].

Definition 2 [39] *A* fence design *is a push plan* $\alpha_1, \ldots, \alpha_k$ *satisfying for all* $1 \leqslant i < k$*:*

$$\begin{aligned} &\alpha_i \in (0, \pi/2) \cup (-\pi, -\pi/2) \Rightarrow \alpha_{i+1} \in (0, \pi/2) \cup (\pi/2, \pi) \\ \wedge\ &\alpha_i \in (-\pi/2, 0) \cup (\pi/2, \pi) \Rightarrow \alpha_{i+1} \in (-\pi/2, 0) \cup (-\pi, -\pi/2). \end{aligned}$$

Definition 2 immediately shows that the linear-length push plans by Chen and Ierardi are valid fence designs for parts with a push function with a uniquely



(a)

t_i	α_{i+1}	t_{i+1}
left	$(0, \pi/2)$	left
left	$(\pi/2, \pi)$	right
right	$(-\pi, -\pi/2)$	left
right	$(-\pi/2, 0)$	right

(b)

Fig. 7. (a) For two successive left fences, the reorientation of the push direction lies in the range $(0, \pi/2)$. (b) The ranges op possible reorientations of the push direction for all pairs of successive fence types.

defined longest half-step of length $\alpha < \pi/2$. In other words, such parts can be oriented by a sequence of equivalent fences along one side of the belt of length $O(n)$. It is much harder to prove that all other parts can also be oriented by a sequence of fences [8].

Theorem 2. *Any polygonal part with n vertices can be oriented up to symmetry by a fence design.*

The results from the preceding section indicate that eccentric parts can be oriented by a constant number of fences.

3.1 A simple graph-based algorithm

We now turn our attention to the computation of the shortest fence design that will orient a given part. We denote the sequence of *stable* equilibrium orientations of P by Σ. As every fence puts the part in a stable equilibrium orientation, the part is in one of these $|\Sigma| = O(n)$ orientations as it travels from one fence to another. Let us label these stable equilibria $\sigma_1, \ldots, \sigma_{|\Sigma|}$. The problem is to reduce the set of possible orientations of P to one stable equilibrium $\sigma_i \in \Sigma$ by a sequence of fences. We build a directed graph on all possible *states* of the part as it travels from one fence to a next fence. A state consists of a set of possible orientations of the part plus the type (left or right) of the last fence, as the latter imposes a restriction on the reorientation of the push direction. Although there are $2^{|\Sigma|}$ subsets of Σ, it turns out that we can restrict ourselves to subsets consisting of sequences of adjacent stable equilibria. Any such sequence can be represented by a closed interval I of the form $[\sigma_i, \sigma_j]$ with $\sigma_i, \sigma_j \in \Sigma$. The resulting graph has $|\Sigma|^2$ nodes.

Consider two graph nodes (I, t) and (I', t'), where $I = [\sigma_i, \sigma_j]$ and I' are intervals of stable equilibria and t and t' are fence types. Let $A_{t,t'}$ be the open interval of reorientations admitted by the successive fences of types t and t'

according to Figure 7(b). There is a directed edge from (I,t) to (I',t') if there is an angle $\alpha \in A_{t,t'}$ such that a reorientation of the push direction by α followed by a push moves any stable equilibrium in I into a stable orientation in I'. To check this condition, we determine the preimage $(\phi,\psi) \supseteq I'$ of I' under the push function. Observe that if $|I| = \sigma_j - \sigma_i < \psi - \phi$, any reorientation in the open interval $(\phi - \sigma_i, \psi - \sigma_j)$ followed by a push will map I into I'. We add an edge from (I,t) to (I',t') if $(\phi - \sigma_i, \psi - \sigma_j) \cap A_{t,t'} \neq \emptyset$, and label this edge with this non-empty intersection. For convenience, we add a source and a sink to the graph. We connect the source to every node $(I = [\sigma_i, \sigma_{i-1}], t)$, and we connect every node $(I = [\sigma_i, \sigma_i], t)$ to the sink. The graph has $O(n^4)$ edges. Every path from the source to the sink now represents a fence design. A fence design of minimum length corresponds to a shortest such path.

An important observation is that some graph edges are redundant if we are just interested in a fence design of minimum length. Consider a node (I,t) and all its outgoing edges to nodes $(I' = [\sigma_i, \sigma_j], t')$ for a fixed σ_I and t'. Lemma 3 [11] shows that only the edge to the node corresponding to the shortest such I' is required.

Lemma 3. *Let (I,t), (I',t'), and (I'',t') be nodes such that I' and I'' have a common left endpoint, and $I' \subset I''$. If there are edges from (I,t) to both (I',t') and (I'',t') then the edge from (I,t) to (I'',t') can be deleted without affecting the length of the shortest path.*

Informally, the lemma says that we can afford to be greedy in our wish to reduce the length of the interval of possible orientations. It reduces the number of graph edges to $O(n^3)$.

The computation of the reduced graph for fence design is easy. In the reduced graph, each node with interval $[\sigma_i, \sigma_j]$, has just one outgoing edge to the set of nodes with intervals with a common left endpoint $\sigma_{i'}$ and a common fence type t'. The shortest interval with left endpoint $\sigma_{i'}$ is obtained by a push direction which maps σ_i onto $\sigma_{i'} - \ell$, where ℓ is the length of the half-step left of $\sigma_{i'}$. The construction of the graph follows directly from this observation. We align the interval with the left environment of the reachable orientations for a valid reorientation of the push direction, and compute the resulting interval after application of the push function. If it is not possible to align σ_i with $\sigma_{i'} - \ell$, then we take the reorientation of the jaw that gets us as close as possible to $\sigma_{i'} - \ell$.

The computation of the outgoing edges for one node can be accomplished in linear time, by shifting $[\sigma_i, \sigma_j]$ along the possible reorientations of the push direction. As a result, the total time required to compute the graph edges is $O(n^3)$. A breadth-first search on the graph takes $O(n^3)$ time, and results in the shortest fence design.

Theorem 3. *Let P be a polygonal part with n vertices. The shortest fence design that orients p up to symmetry can be computed in $O(n^3)$ time. The resulting design consists of $O(n^2)$ fences in the worst case.*

Theorem 3 immediately provides an upper bound of $O(n^2)$ on the length of the shortest fence design. We expect, however, that the true bound is $O(n)$.

3.2 An output-sensitive algorithm

The running time of the preceding algorithm could be considered quite high when realizing that fence designs will often (or maybe even always) have linear (in the case of parts push functions with a unique longest half-step) or even constant (in the case of eccentric parts) length. This suggests that an algorithm whose running time is sensitive to the length of the fence design is to be preferred.

The main idea of the output-sensitive algorithm is to maintain the shortest interval of possible orientations after k fences, instead of precomputing the whole graph of all possible intervals of orientations. This is basically the same technique as used by Goldberg's algorithm to compute push plans [26]. Goldberg maintains the interval of possible orientations, and greedily shrinks this interval per application of the pushing jaw. We, however, must take into account the constraints of fence design. It is not sufficient to maintain a single shortest interval of possible orientations. Lemma 3 indicates that it is sufficient to maintain for each pair of a fence type and a stable orientation the shortest interval after leaving a fence of the given type starting with the given stable orientation. The algorithm should terminate as soon as one of the $2|\Sigma|$ intervals has shrunk to a single orientation. Updating the candidate intervals can be accomplished in $(\log n)$ time per interval using a range tree data structure (see [10] for details).

Theorem 4. *Let P be a polygonal part with n vertices. A shortest fence design that orients P up to symmetry can be computed in $O(kn \log n)$ time, where k is the length of the resulting fence design.*

The output-sensitive algorithm will in most cases be more efficient than the simpler graph-based approach; in fact, the former will only have a chance to be outperformed by the latter if parts exist that require a quadratic-length fence design.

Both algorithms can be modified to deal with situations in which there is friction between the part and the fences. This modification has no impact on the running time. On the other hand we lose the guarantee that a fence design always exists, so that the algorithm may have to report failure. The output-sensitive algorithm will be able to do so in $(n^3 \log n)$ time. See [10, 8] for other extensions.

4 Pushing three-dimensional parts

A drawback of most achievements in the field of sensorless orientation of parts is that they only apply to planar parts, or to parts that are known to rest on a certain face. The generalization of conveyor belts and fences that we describe here attempts to bridge the gap to truly three-dimensional parts. The device we use is a cylinder with plates tilted toward the interior of the cylinder attached to the side. Across the plates there are fences. The part cascades down from plate to plate, and slides along the fences as it travels down a plate (see Figure 8(a)). The plate on which the part slides discretizes the first two degrees of freedom of

rotation of the part. A part in alignment with a plate retains one undiscretized rotational degree of freedom. The orientation of the part is determined up to its roll, i.e. the rotation about the axis perpendicular to the plate. The fences, which are mounted across the plates, push the part from the side, and discretize the roll. We assume that P first settles on the plate before it reaches the fences which are mounted across the plate. Moreover, we assume that the fences do not topple the part but only cause it to rotate about the roll axis.

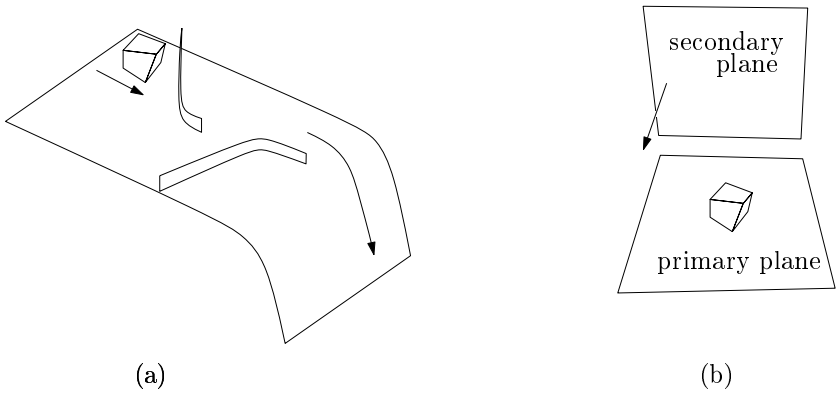


Fig. 8. (a) A part sliding down a plate with fences. (b) The same part on the jaw.

The objective of this section is to compute a set-up of plates and fences that is guaranteed to move a given asymmetric polyhedral part towards a unique final orientation. Such a set-up, or design, consists of a sequence of plate slopes, and for each plate a sequence of fence orientations.

When a part moves along a fence on a plate, it is essentially pushed from two orthogonal directions. This motivates us to first study the fundamental (but artificial) problem of pushing in three-dimensional space. Here, the part is assumed to float in the air while we push it from two orthogonal directions.

We show that a three-dimensional polyhedral part P can be oriented up to symmetry by a (particular) sequence of push actions, a *push plan*, of length $O(n^2)$, where n is the number of vertices of P. Furthermore, we give an $O(n^3 \log n)$ time algorithm to compute such a push plan. We show how to transform this three-dimensional push plan to a three-dimensional design for the plates and fences. The resulting design consists of $O(n^3)$ plates and fences, and can be computed in $O(n^4 \log n)$ time.

A polyhedral part in three-dimensional space has three rotational degrees of freedom. We assume that a fixed reference frame is attached to P and denote the orientation of P relative to this reference frame by (ϕ, ψ, θ), where (ϕ, ψ) denotes a point on the sphere of directions, and θ is the roll about the ray emanating from the origin through (ϕ, ψ).

4.1 Push plan

We study the push actions of the plates and the fences in a more general setting by replacing a plate and a fence by two orthogonal planes. We call the planes the primary and secondary plane, respectively. A picture of the resulting jaw is given in Figure 8(b). Since the planes can only touch P at its convex hull, we assume without loss of generality that P is convex. We assume that the center-of-mass of P, denoted by c, is in the interior of P. Analogously to the cylindrical feeder, we assume that only after P has aligned with the primary plane, we apply the secondary plane. As the part rests on the primary plane, the secondary plane pushes P at its orthogonal projection onto the primary plane. We assume that the feature on which P rests retains contact with the primary plane as the secondary plane touches P. We assume that for any equilibrium orientation, which is an orientation for which P rests on the jaw, the projection of P onto the primary plane has no symmetry. We refer to a part with this property as being *asymmetric*.

In order to be able to approach the part from any direction, we make the (obviously unrealistic) assumption that the part floats in the air, and assume that we can control some kind of gravitational field which attracts the part in a direction towards the jaw. Also, we assume that the part quasi-statically aligns with the jaw, meaning that we ignore inertia.

A basic action of the jaw consists of directing and applying the jaw. The result of a basic action for a part in its reference orientation is given by the *push function*. The push function $p : [0, 2\pi) \times [-\pi/2, \pi/2] \times [0, 2\pi) \rightarrow [0, 2\pi) \times [-\pi/2, \pi/2] \times [0, 2\pi)$ maps a push direction of the jaw relative to P in its reference orientation onto the orientation of P after alignment with the jaw. The orientation of P after a basic action for a different initial orientation than its reference orientation is equal to the push function for the push direction plus the offset between the reference and the actual initial orientation of P.

In our approach to finding a push plan we do not explicitly compute the push function. Instead we occasionally query some data structure for the reorientation of the part when being pushed from a certain direction. Without going into the details, which are far from easy, we claim that this query takes $O(n \log n)$ time. We now use this fact to show that any asymmetric polyhedral part P can be oriented by a push plan of length $O(n^2)$. The part P has at most $O(n)$ equilibria with respect to the primary plane, and any projection of P onto the primary plane has $O(n)$ vertices. Hence, the total number of orientations of P compliant to the jaw is $O(n^2)$, and this bound turns out to be tight.

Let us, for a moment, assume that the part lies in a stable orientation on the primary plane. We can now reorient the jaw in such a way that the contact direction of the primary plane remains unchanged while the direction of the secondary plane is altered. A subsequent push by the jaw will cause the part to rotate about the normal to the pimary plane—keeping the same face of P in contact with the primary plane. The application of the jaw in this manner can therefore be regarded as a push operation on the 2D orthogonal projection of P. In Section 2 we have seen that an asymmetric 2D part with m vertices can

be oriented up by means of planar push plan of length $O(m)$. Consequently, we can orient P in stable contact with the primary plane by $O(n)$ applications of the secondary plane.

Lemma 4. *Let P be an asymmetric polyhedral part with n vertices. There exists a plan of length $O(n)$ that puts P into a given orientation (ϕ, ψ, θ) from any initial orientation (ϕ, ψ, θ')*

We call the operation that orients P for a single stable equilibrium contact direction (ϕ, ψ) of the primary plane CollideRollsSequence(ϕ, ψ). It allows us to eliminate the uncertainty in the roll for any stable contact direction of the primary plane. In an initialization phase we reduce the number of possible orientations of P to $O(n)$ by executing CollideRollsSequence(ϕ, ψ) for all equilibrium contact directions (ϕ, ψ) of the primary plane. We let Σ be the set of the resulting possible orientations. Lemma 5 (see [15] for a proof) provides us with push operations to further reduce the number of possible orientations.

Lemma 5. *There exist two antipodal reorientations of the primary plane that map any pair of orientations (ϕ, ψ, θ), and (ϕ', ψ', θ') of a polyhedral part onto orientations $(\tilde{\phi}, \tilde{\psi}, \tilde{\theta})$ and $(\tilde{\phi}', \tilde{\psi}', \tilde{\theta}')$ that satisfy $\tilde{\phi} = \tilde{\phi}'$ and $\tilde{\psi} = \tilde{\psi}'$.*

We call the basic operation that collides two orientations onto the same equilibrium for the primary plane CollidePrimaryAction. Combining Lemma 4 and 5 leads to a construction of a push plan for a polyhedral part. The following algorithm orients a polyhedral part without symmetry in the planar projections onto supporting planes of its stable faces.

OrientPolyhedron(P):
▷ After initialization $|\Sigma| = O(n)$
1. **while** $|\Sigma| > 1$ **do**
   2.1 pick (ϕ, ψ, θ), $(\phi', \psi', \theta') \in \Sigma$
   2.2 plan $\leftarrow$ CollidePrimaryAction$((\phi, \psi, \theta), (\phi', \psi', \theta'))$
   ▷ Lemma 5;
   ▷ plan$(\phi, \psi, \theta) = (\phi'', \psi'', \theta'')$, and plan$(\phi', \psi', \theta') = (\phi'', \psi'', \theta''')$
   2.3 **for all** $(\tilde{\phi}, \tilde{\psi}, \tilde{\theta}) \in \Sigma$
   2.3.1 $(\tilde{\phi}, \tilde{\psi}, \tilde{\theta}) \leftarrow$ plan$(\tilde{\phi}, \tilde{\psi}, \tilde{\theta})$.
   2.4 plan $\leftarrow$ CollideRollsSequence(ϕ'', ψ'')
   ▷ Lemma 4
   2.5 **for all** $(\tilde{\phi}, \tilde{\psi}, \tilde{\theta}) \in \Sigma$
   2.5.1 $(\tilde{\phi}, \tilde{\psi}, \tilde{\theta}) \leftarrow$ plan$(\tilde{\phi}, \tilde{\psi}, \tilde{\theta})$.

The algorithm repeatedly takes two of the remaining possible orientations of the part and computes a reorientation that maps these two orientations onto two different orientations whose representations share the first two coordinates. Step 2.3 maps all currently possible orientations onto the orientations result from applying the appropriately reoriented jaw. We recall that this step takes $O(n \log n)$ for each of the at most $O(n)$ remaining orientations. At this stage,

the number of faces of P that can be aligned with the primary plane is reduced by one. The remaining steps map the two orientations that share the first two coordinates onto a single orientation—essentially by means of a planar push plan of $O(n)$ length for the projection of P. Since the number of iterations of ORIENTPOLYHEDRON(P) is $O(n)$ the algorithm runs in $O(n^3 \log n)$ time and results in a push plan of length $O(n^2)$.

Theorem 5. *A push plan of length $O(n^2)$ for an asymmetric polyhedral part with n vertices can be computed in $O(n^3 \log n)$ time.*

4.2 Plates and fences

We use the results from the preceding subsection to determine a design for the feeder consisting of tilted plates with curved tips, each carrying a sequence of fences. The motion of the part effectively turns the role of the plates into the role of the primary pushing plane, and the role of the fences into the role of the secondary pushing plane. We assume that the part quasi-statically aligns to the next plate, similar to the alignment with the primary plane of the generic jaw. Also, we assume that the contact direction of the plate does not change as the fences brush the part, i.e. the part does not tumble over.

The fact that the direction of the push, i.e., the normal at the fence, must have a non-zero component in the direction opposite to the motion of the part, which slides downward under the influence of gravity, imposes a restriction on successive push directions of the secondary plane. The restriction is equivalent to that in planar fence design. Theorem 3 shows that it is possible to orient a planar polygonal part (hence a polyhedral part resting on a fixed face) using $O(n^2)$ fences. The optimal fence design can be computed in $O(n^3)$ time.

As the part moves towards the end of a plate, the curved end of the plate causes the feature on which the part rests to align with the vertical axis, while retaining the roll of the part. When the part leaves the plate, the next plate can only push the part from below. This draws restrictions on the possible reorientations of the primary plane, in the model with the generic three-dimensional jaw (see Figure 9). Careful analysis shows that the reorientation of the primary plane is within $(-\pi, 0) \times (0, \pi)$ when the last fence of the last plate was a left fence. Similarly, for a last right fence, the reorientation of the primary plane is within $(0, \pi) \times (0, \pi)$.

The gravitational force restricts our possible orientations of the primary plane in the general framework. Fortunately, Lemma 5 gives us two antipodal possible reorientations of the primary plane. It is not hard to see that one of these reorientations is in the reachable hemisphere of reorientations of the push direction of the primary plane for two succesive plates. This implies we can still find a set-up of plates and fences of $O(n^3)$ complexity.

Theorem 6. *An asymmetric polyhedral part can be oriented using $O(n^3)$ fences and plates. We can compute the design in $O(n^4 \log n)$ time.*

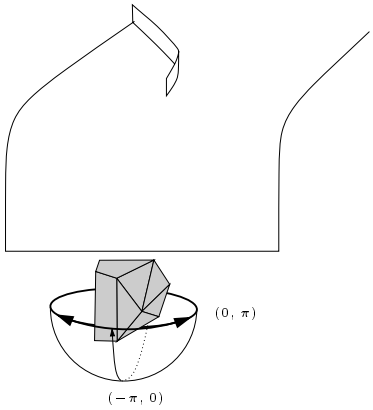


Fig. 9. The next plate can only touch the lower half of the part.

5 Trap design

The oldest and still most common part feeder is the *vibratory bowl feeder*. It consists of a bowl filled with parts surrounded by a helical metal track [18, 19]. 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 and traps. 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. In this section, we consider the use of traps to filter polygonal parts on a track. A trap is a (partial) interruption of the track. We focus on polygonal traps. Figure 10 shows a section of track with a rectangular trap. Parts in undesired orientations fall back into the bowl, other orientations

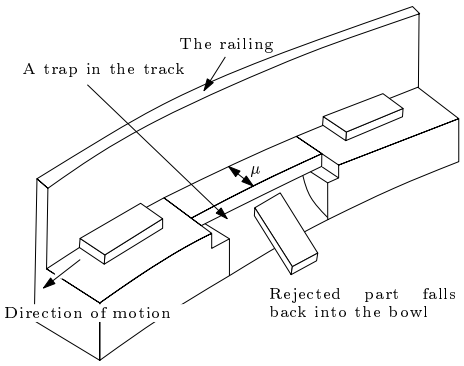


Fig. 10. Vibratory bowl feeder track [19].

remain supported.

Specific to vibratory bowls, researchers have used simulation [7, 27, 32], heuristics [28], and genetic algorithms [24] to design traps. Perhaps closest in spirit to our work is Caine's PhD thesis [21] which develops geometric analysis tools to help designers by rendering the configuration-space for a given combination of part, trap, and obstacle. Caine also gives some heuristics to design feeder track features.

This section reports on the analysis and design of traps that allow a part to pass in only one orientation. To the extent of our knowledge, no research in the systematic algorithmic design of vibratory bowl feeders has previously been conducted. As the techniques and analyses used in trap design differ largely from those used for the feeders in the preceding sections—which are all based on pushing—we do not provide extensive coverage of all our algorithms for traps. Instead we confine ourselves to a brief characterization of when a part falls into a trap and to reporting our algorithmic results for trap analysis and design. We focus on two-dimensional parts, or, in other words on three-dimensional parts that are known to rest on a certain stable face.

5.1 Modeling and analysis

The track in a bowl feeder is slightly tilted to keep the part in contact with the railing of the track as it moves. Although the vibration of the bowl causes the part to hop along the track we simplify our analysis by assuming that is slides. The radius function of the part P determines the at most $O(n)$ stable orientations in which the part can move; these correspond to local minima of the radius function.

Let T be a polygonal trap, and assume it has m vertices. In reality, the part P (which is assumed to be in a fixed orientation) slides across the trap. Since is it convenient to have a stationary center-of-mass c of the part in our analysis, we choose to assume that the trap moves underneath the part (which is clearly equivalent). We denote the trap in a configuration q by $T(q)$. Note that a configuration is—because of the simple sliding motion of the part in reality—nothing more than a horizontal displacement, and as such representable by a single value. The supported area $S(q)$ of a part P with the trap placed underneath in configuration q is defined by $S(q) = P - \text{int}(T(q))$. We denote the convex hull of a shape X by $CH(X)$. Lemma 6 says when a part is safe when placed over a trap, i.e., when it does not fall into the trap.

Lemma 6. *The part P is safe above T in a configuration q if and only if $c \in CH(S(q))$, or, in other words, if and only if there is no line through c that has $CH(S(q))$ entirely on one side.*

It is clear that a part will safely move across a trap if P is safe in every configuration q in the motion of T.

The above characterization is the key to our algorithm for computing whether a part P in a given orientation will safely move across a trap. The crucial convex hull $CH(S(q))$ is determined by vertices of T and P, and by intersections of

edges of T and P. Our algorithm plots the directions of the rays emenating from c towards each of the aforementioned vertices and intersections as a function of q. A simple sweep (see e.g. [6]) suffices to detect whether the interval of all these directions remains longer than π at all q. A somewhat more efficient algorithm exists for the case where both the trap and the part are convex.

Theorem 7. *Let P be a polygonal part with n vertices and T be a polygonal trap with m vertices. We can report whether P will move safely across T in $O(n^2 m \log n)$ time, or in $O((n+m)\log n)$ time if both P and T are convex.*

The result for a general part and trap has recently been improved by Agarwal et al. [1].

5.2 Design of traps

In this subsection we report results on the design of traps that allow a given part to pass in only one of its stable orientations. A trap with this property is said to have the feeding property. We consider four different specific rectangular traps and arbitrary polygonal traps. Figure 11 shows the four rectilinear traps along with the parameters that specify their measures.

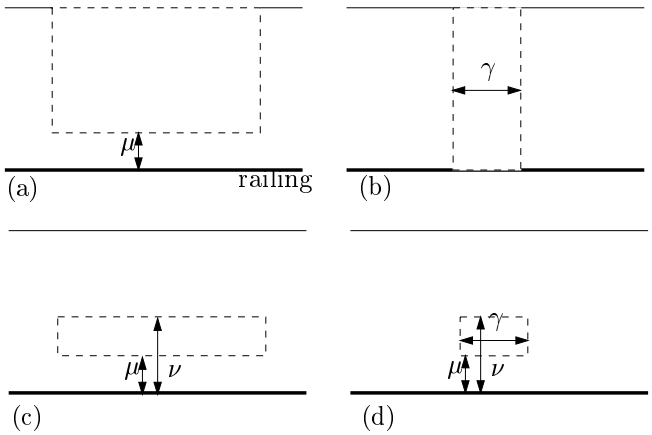


Fig. 11. The four rectilinear traps of this section: (a) a balcony, (b) a gap, (c) a canyon, and (d) a slot. The bold lines at the bottom of the pictures depict the railing. The line at the top depicts the edge of the track at the inside of the bowl. The traps are dashed.

A balcony is a long interruption of the upper part of the supporting area of the track. Let us consider the radii of the part P, or, in other words, the distances of the center-of-mass c to the railing in all of its stable orientations. Assume that there is a uniquely defined minimum, corresponding to an orientation ϕ. If we

choose the height μ of the balcony slightly larger than this minimum, then it is immediately clear that P will only be able to pass T when in orientation ϕ. As the minimum radius is computable in $O(n)$ time we obtain the following result.

Theorem 8. *In $O(n)$ time we can design a balcony with the feeding property for a polygonal part with n vertices, or report that no such balcony exists.*

Unfortunately, the design of the other feeders is considerably harder.

A gap is an interruption of the trach that spans the entire width of the track. Its shape is determined by a single parameter, the gap length γ. Our algorithm (see [9, 12, 13] for details) determines a choice for γ that allows P to pass in only one orientation.

Theorem 9. *In $O(n^2 \log n)$ time we can design a gap with the feeding property for a polygonal part with n vertices, or report that no such gap exists. The bound reduces to $O(n^2)$ if the part is convex.*

A canyon is a long rectangular interruption of the supporting area of the track. Its shape is defined by the distances μ and ν from the lower and upper boundary to the railing. Our algorithm [9, 12, 13] determines a suitable choice for μ and ν.

Theorem 10. *In $O(n^2 \alpha(n) \log n)$ time we can design a canyon with the feeding property for a polygonal part with n vertices, or report that no such canyon exists; $\alpha(n)$ is the extremely slowly growing inverse Ackermann function.*

A slot is a true rectangular interruption of the supporting area of the track, and as such specified by three parameters μ, ν, and γ. Our algorithm [9, 12, 13] finds a slot with the feeding property if one exists.

Theorem 11. *In $O(n^8)$ time we can design a slot with the feeding property for a polygonal part with n vertices, or report that no such slot exists.*

Finally, we consider arbitrary polygonal traps with k vertices. Such a trap can be represented by $2k$ parameters. Our approach to computing a k-vertex trap that allows a given part to pass in only one orientation uses high-dimensional arrangements and quantifier elimination. Using recent results by Basu et al. [4, 5], we obtain our final result [12], which is given below.

Theorem 12. *In $O((nk)^{O(k^2)})$ time we can design a polygonal trap with k vertices with the feeding property for a polygonal part with n vertices, or report that no such trap exists.*

References

1. P.K. Agarwal, A.D. Collins, and J.L. Harer. Minimal trap design. 2000.
2. S. Akella, W. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26:313–344, 2000.

3. S. Akella and M. T. Mason. Posing polygonal objects in the plane by pushing. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2255–2262, 1992.
4. S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM*, 46(4):537–555, 1999.
5. S. Basu, R. Pollack, and M-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43:1002–1045, 1996.
6. M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
7. D. Berkowitz and J. Canny. Designing parts feeders using dynamic simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1127–1132, 1996.
8. R-P. Berretty. *Geometric design of part feeders*. PhD thesis, Institute of Information and Computing Sciences, Utrecht University, 2000.
9. R-P. Berretty, K. Y. Goldberg, L. Cheung, M. H. Overmars, G. Smith, and A. F. van der Stappen. Trap design for vibratory bowl feeders. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2558–2563, 1999.
10. R-P. Berretty, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Algorithms for fence design. In *Robotics, the algorithmic perspective*, pages 279–295. A.K. Peters, 1998.
11. R-P. Berretty, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Computing fence designs for orienting parts. *Computational Geometry: Theory and Applications*, 10(4):249–262, 1998.
12. R-P. Berretty, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Geometric techniques for trap design. In *Annual ACM Symposium on Computational Geometry*, pages 95–104, 1999.
13. R-P. Berretty, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Geometric trap design for automatic part feeders. In *International Symposium on Robotics Research (ISRR)*, pages 139–144, 1999.
14. R-P. Berretty, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Orienting parts by inside-out pulling. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2001. To appear.
15. R-P. Berretty, M. H. Overmars, and A. F. van der Stappen. Orienting polyhedral parts by pushing. *Computational Geometry: Theory and Applications*, 2001. To appear.
16. K-F. Böhringer, V. Bhatt, B.R. Donald, and K. Y. Goldberg. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26:389–429, 2000.
17. K-F. Böhringer, B. R. Donald, and N.C. MacDonald. Upper and lower bounds for programmable vector fields with applications to mems and vibratory plate part feeders. *Algorithms for Robotic Motion and Manipulation,* J.-P. Laumond and M. Overmars (Eds.), A.K. Peters, pages 255–276, 1996.
18. G. Boothroyd and P. Dewhurst. *Design for Assembly – A Designers Handbook*. Department of Mechanical Engineering, University of Massachusetts, Amherst, Mass., 1983.
19. G. Boothroyd, C. Poli, and L. Murch. *Automatic Assembly*. Marcel Dekker, Inc., New York, 1982.
20. M. Brokowski, M. A. Peshkin, and K. Y. Goldberg. Optimal curved fences for part alignment on a belt. *ASME Transactions of Mechanical Design*, 117, 1995.
21. M. E. Caine. The design of shape interaction using motion constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 366–371, 1994.

22. J. Canny and K. Y. Goldberg. Risc for industrial robotics: Recent results and open problems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1951–1958, 1994.
23. Y-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14:367–397, 1995.
24. 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)*, pages 846–851, 1996.
25. M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, 4:367–379, 1988.
26. K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2):201–225, 1993.
27. M. Jakiela and J. Krishnasamy. Computer simulation of vibratory parts feeding and assembly. In *International Conference on Discrete Element Methods*, pages 403–411, 1993.
28. 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):2513–2524, 1994.
29. K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, 15(6):533–556, 1996.
30. M. T. Mason. Mechanics of robotic manipulation. Unpublished book.
31. M. T. Mason. *Manipulator grasping and pushing operations.* PhD thesis, MIT, 1982. published in *Robot Hands and the Mechanics of Manipulation*, MIT Press, Cambridge, 1985.
32. G. Maul and M. Thomas. A systems model and simulation of the vibratory bowl feeder. *Journal of Manufacturing Systems*, 16(5):309–314, 1997.
33. B. K. Natarajan. Some paradigms for the automated design of parts feeders. *International Journal of Robotics Research*, 8(6):89–109, 1989.
34. M. A. Peshkin and A. C. Sanderson. The motion of a pushed sliding workpiece. *IEEE Journal of Robotics and Automation*, 4(6):569–598, 1988.
35. M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, pages 696–701, 1988.
36. A. Rao and K. Y. Goldberg. Manipulating algebraic parts in the plane. *IEEE Transactions on Robotics and Automation*, 11:589–602, 1995.
37. A. F. van der Stappen, K. Y. Goldberg, and M. H. Overmars. Geometric eccentricity and the complexity of manipulation plans. *Algorithmica*, 26:494–514, 2000.
38. D. E. Whitney. Real robots don't need jigs. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 746–752, 1986.
39. J. A. Wiegley, K. Y. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. *Assembly Automation*, 17(2):129–136, 1997.