Robert-Paul Berretty

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175, USA

Ken Goldberg

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

Mark H. Overmars
A. Frank van der Stappen

Institute of Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands

Trap Design for Vibratory Bowl Feeders

Abstract

The vibratory bowl feeder is the oldest and still most common approach to the automated feeding (orienting) of industrial parts. In this paper, the authors consider a class of vibratory bowl filters that can be described by removing polygonal sections from the track; this class of filters is referred to as traps. For an n-*sided polygonal part and an* m-*sided polygonal trap, an* $O(n^2 m \log n)$ *algorithm is given to decide whether the part in a specific orientation will safely move across the trap or will fall through the trap and thus be filtered out. For an* n-*sided convex polygonal part and* m-*sided convex polygonal trap, this bound is improved to* $O((n+m) \log n)$. *Furthermore, the authors show how to design various trap shapes, ranging from simple traps to general polygons, which will filter out all but one of the different stable orientations of a given part. Although the runtimes of the design algorithms are exponential in the number of trap parameters, many industrial part feeders use few-parameter traps (balconies, canyons, slots); in these cases, the running times of the algorithms range from linear to low-degree polynomial.*

KEY WORDS—sensorless orientation, vibratory bowl feeder, trap design, computational geometry

1. Introduction

A *part feeder* takes in a stream of identical parts in arbitrary orientations and outputs them in a uniform orientation. We consider the problem of *sensorless orientation* of parts, in which the initial orientation of the part is assumed to be

unknown. In sensorless manipulation, parts are positioned and/or oriented using passive mechanical compliance. The input is a description of the part shape, and the output is a sequence of open-loop actions that move a part from an unknown initial orientation into a unique final orientation. Among the sensorless part feeders considered in literature are the parallel-jaw gripper (Chen and Ierardi 1995; Goldberg 1993), the single pushing jaw (Akella and Mason 1992; Lynch and Mason 1996; Mason 1982; Peshkin and Sanderson 1988a), the conveyor belt with a sequence of (stationary) fences placed along its sides (Berretty et al. 1998; Brokowski, Peshkin, and Goldberg 1995; Peshkin and Sanderson 1988b; Wiegley et al. 1997), the conveyor belt with a single rotational fence (Akella et al., 2000), the tilting tray (Erdmann and Mason 1988; Natarajan 1989), vibratory plates, and programmable vector fields (Böhringer et al. 2000).

The oldest and still most common approach to automated feeding is the *vibratory bowl feeder*. It consists of a bowl filled with parts surrounded by a helical metal track (Boothroyd, Poli, and Murch 1982; Boothroyd and Dewhurst 1983). 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. Thus, a stream of oriented parts emerges at the top after successfully running the gauntlet. In this paper, we present a framework to filter polygonal parts on a track using traps. A trap is described by removing polygonal sections from the track. A picture of a section of the feeder track is given in Figure 1. The parts move from the right to
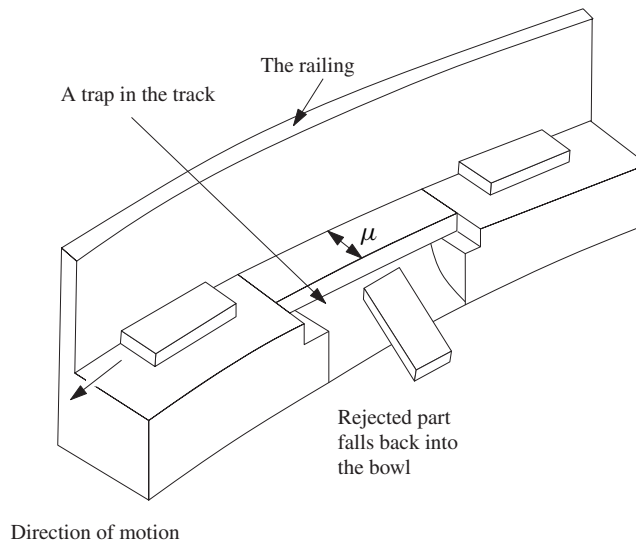
Fig. 1. Vibratory bowl feeder track (Boothroyd, Poli, and Murch 1982).

the left on the feeder track. Parts in undesired orientations fall back into the bowl; other orientations remain supported.

Vibratory bowl feeders "are the versatile, rugged, day-in, day-out workhorses that shoulder the bulk of all automatic part feeding operations" (Riley 1983). Chapter 5 of Riley's handbook provides a detailed taxonomy of industrial part feeding mechanisms and notes that bowl design currently relies on human trial and error and often requires substantial debugging on the assembly line. Specific to these vibratory bowls, researchers have used simulation (Berkowitz and Canny 1996; Jakiela and Krishnasamy 1993; Maul and Thomas 1997), heuristics (Lim et al. 1994), and genetic algorithms (Christiansen, Edwards, and Coello 1996) to design traps. Perhaps closest in spirit to our work is Caine's (1994) Ph.D. thesis, 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. In this paper, we do not propose heuristics or user-guided design to define traps but rather give algorithms that are guaranteed to always find a trap of a preferred shape, if such a trap exists. Very recent results furthering our geometric approach of trap design were reported in a paper by Agarwal, Collins, and Harer (2001), who observed that the running time of our algorithm in Section 3 can be improved. Moreover, they show how to find a family of part–geometry derived shapes, if such trap shapes exist.

Consider a part feeding system that accepts as input a set of part orientations Σ. Based on a definition by Akella et al. (2000), we might say that a system has the *feeding property* if there exists some orientation σ, usually in Σ, such that the system outputs parts only in orientation σ. This paper reports

on algorithms that design traps with the feeding property. To the extent of our knowledge, these are the first results in the systematic design of feeder traps.

This paper is organized as follows. In Section 2, we give a geometric model of the bowl feeder; this model is the basis for our algorithms. In Section 3, we analyze whether a part in a given orientation will safely move across the trap or will be filtered out and fall back into the bowl. For a polygonal part with n vertices and a polygonal trap with m vertices, the resulting algorithm runs in $O(n^2 m \log n)$ time. This can be improved to $O((n+m) \log n)$ time if the part and the trap are convex. In Section 4, we give algorithms for designing traps in the feeder track. We construct, for example, a gap that is a rectilinear interruption of the track. Given the geometry of the part, we compute in $O(n^2 \log n)$ time how long the gap should be to establish a feeder. This bound is reduced to $O(n^2)$ for convex parts. We also consider other trap shapes— the balcony, the canyon, and the slot—and conclude with a general approach for designing general polygonal traps. Several algorithms of our paper have been implemented, and the resulting traps have been successfully tested in an experimental setup. The experiments, accompanied by a discussion of the assumptions implied by our geometric model, are reported in Section 5. We conclude this paper in Section 6.

2. Preliminaries

In this section, we discuss the geometric properties of the bowl feeder. The geometric model we will present is designed to reflect important characteristics of the bowl feeder. In Section 5, we describe experiments with a real feeder that uses

traps generated with our model and the discrepancies between our model and part feeding in practice.

In this paper, we address the problem in the plane. Throughout this paper, P denotes a two-dimensional polygonal part. The two-dimensional polygonal trap in the track is denoted by T. The number of vertices of P is denoted by n. The number of vertices of T is denoted by m.

A subset S of the plane is called *convex* if and only if for any pair of points $p, q \in S$, the line segment (p, q) is completely contained in S. The *convex hull*, $\mathcal{CH}(S)$, of a set S is the smallest convex set that contains S. We denote the interior of a set S by $\text{int}(S)$.

The part has a center of mass c, which lies inside the convex hull of the part. At c, a fixed coordinate frame is attached, which identifies the zero orientation of the part. The track is slightly tilted toward the railing so the part remains in contact with the railing as it moves along the railing. The radius function for the part characterizes the stable orientations of the part against the railing (Goldberg 1993).

DEFINITION 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 a part can easily be computed in linear time from the description of the part (Mason 1982). The orientation of a part is identified by the angle between the reference frame and the y-axis. In our model, the possible orientation of the part is restricted to $O(n)$ different, stable orientations.

In reality, the part is mobile and slides across a stationary trap in the positive x-direction. It is, however, easier to describe the solutions by viewing the part as stationary and to slide the trap underneath the part (which is obviously equivalent). We assume that the railing of the track is aligned with the x-axis. Throughout the motion of the trap, c is on the y-axis at a fixed distance c_y from the railing, and the part's orientation does not change.

All figures in this paper have the railing coincident with the horizontal axis, and the trap is supposed to move in the negative x-direction. The railing is depicted at the bottom of the figures (see, e.g., Fig. 2).

A placement of the trap (i.e., its horizontal displacement) is denoted by a single value, q. We denote the set of points of the plane covered by trap T at placement q by $T(q)$. The supported area of the part above a trap at placement q is $S(q) = P - \text{int}(T(q))$.

We define how to decide whether a part above a trap in a given placement will fall into the bowl or remain safely on the track. The following definition states that a part is safe if there are three points in P surrounding the center of mass that are supported.

DEFINITION 2.   Let P be a part with center of mass c. Let T be a trap. The part P is *safe* above the trap at placement q if and only if there exists a triangle Δ_{t_1, t_2, t_3} with $c \in \Delta_{t_1, t_2, t_3}$ and $t_1, t_2, t_3 \in S(q)$. Otherwise, the part is *unsafe*.

The following lemmas give us easy ways to decide whether the part is safe.

LEMMA 1. P above $T(q)$ is safe if and only if $c \in \mathcal{CH}(S(q))$.

Proof. ($\Rightarrow$) Let P be safe. There is a triangle Δ_{t_1, t_2, t_3}, with $t_1, t_2, t_3 \in S(q)$ and $c \in \Delta$. Clearly, $t_1, t_2, t_3 \in \mathcal{CH}(S(q))$, and consequently, we have $c \in \mathcal{CH}(S(q))$.

($\Leftarrow$) $c \in \mathcal{CH}(S(q))$. We construct a triangle by computing a triangulation of $\mathcal{CH}(S(q))$. Clearly, there is one triangle in the triangulation that contains c. Furthermore, the vertices of $\mathcal{CH}(S(q))$ are in $S(q)$.   □

LEMMA 2. P above $T(q)$ is safe if and only if there is no line ℓ through c with $\mathcal{CH}(S(q))$ in the open half plane defined by ℓ.

Proof. Follows immediately from the previous lemma because $c \in \mathcal{CH}(S(q))$.   □

A *critical placement* of the trap is a placement where c lies on the boundary of $\mathcal{CH}(S(q))$. It follows from Lemma 2 that a critical placement can also be characterized by a line through c that touches the boundary of $\mathcal{CH}(S(q))$ that bounds a half place containing $S(q)$. The following lemma gives a third way to characterize a critical placement.

LEMMA 3.   Let P be a part above a trap $T(q)$. Let c be not in the interior of $\mathcal{CH}(S(q))$. Let ρ_l and ρ_r be two rays emanating from c. Let the left side of ρ_l be tangent to $\mathcal{CH}(S(q))$, and let the right side of ρ_r be tangent to $\mathcal{CH}(S(q))$. The placement of T is critical if and only if the angle between ρ_l and ρ_r is π, or c is a vertex of $\mathcal{CH}(S(q))$.

Since c lies in the interior of P, we note that if T is convex, c never is a vertex of $\mathcal{CH}(S(q))$. Figure 2 depicts a safe part and a trap together with the convex hull of the supported surface.

The notion of safeness gives us a tool to formalize whether a part in a given orientation will survive a given trap. We assume that if the part is unsafe at some placement of the trap, it is rejected and will fall back into the bowl. For many simple trap shapes, this assumption is justified.

DEFINITION 3.   Let P be a part with center of mass c in a given orientation. Let T be a trap. The part P is *fed* if for all placements q, P is safe above $T(q)$. Otherwise, P is *rejected*.

A trap T has a *critical shape* for orientation σ, if T feeds P in orientation σ and T has at least one critical placement.

The ultimate goal is to find a trap that will feed only one of the possible orientations of P. A trap with this property is said to have the *feeding property* (Akella et al. 2000).
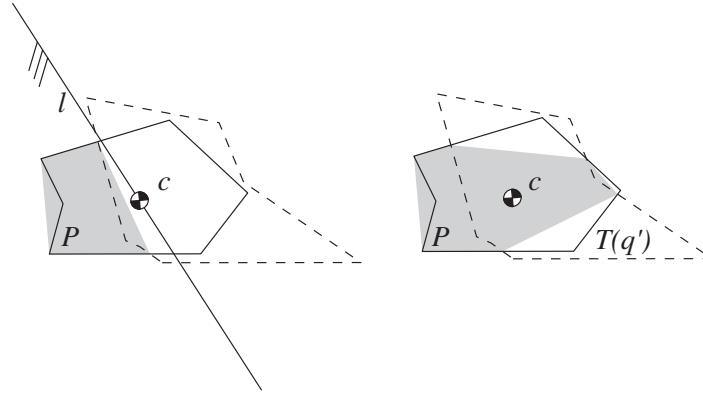
Fig. 2. The part P and its center of mass c above a trap T at different placements q and q'. Placement q corresponds to an unsafe placement. Placement q' is safe. $\mathcal{CH}(S(q))$ and $\mathcal{CH}(S(q'))$ are shaded. The half plane, bounded by line ℓ through c, which contains $\mathcal{CH}(S(q))$, is depicted as well.

3. Analyzing a Trap

In this section, we analyze the safeness of a part above a given trap. In the first subsection, we discuss the general case of a polygonal part above a moving polygonal trap. In the second subsection, we give an algorithm with an improved running time that only deals with convex parts and traps.

3.1. General Polygonal Traps and Parts

In this section, we discuss how to test an orientation of a polygonal part against a polygonal trap in the track. From Section 2, we know that there are at most $O(n)$ different possible orientations for the part on the track. We consider one of these $O(n)$ stable orientations of the part. We answer the question whether P in this specific orientation is fed or rejected. We first give a general algorithm that solves the problem in $O(n^2 m \log n)$ time. Then, we give an improved algorithm that works for convex parts and convex traps and solves the problem in $O((n + m) \log n)$ time.

To determine whether a part will survive a given trap, we sweep the trap underneath the part and check if safeness is retained during the sweep. Lemma 2 gives us the idea of the algorithm. Namely, we check whether at any moment during the sweep, all points of the convex hull of the supported area are in an open half plane through c. If this is not the case, the part is safe.

We distinguish three types of vertices in the arrangement of the two possibly intersecting polygons T and P: the vertices of P, the vertices of T, and the vertices due to the intersections of an edge of P and an edge of T.

The convex hull $\mathcal{CH}(S(q))$ is equal to the convex hull of a subset of these vertices. Recall that $S(q) = P - \text{int}(T(q))$. The vertices of P can only contribute to $\mathcal{CH}(S(q))$ if they are not in T. The trap T does not contribute to $\mathcal{CH}(S(q))$, but we have to take into account the intersection points of edges of P and T. In general, it is not necessary to take into account every intersection between edges of T and edges of P. It is

sufficient, by definition of the convex hull, to only use the (at most) two outermost intersections of each edge of the part.

We compute, for each edge e of the part, the angles of the rays emanating from c through the edge's left- and rightmost point of support during the sweep—for vertical edges, we compute these angles for the lowest and highest point of support. We call these rays *extremal rays*. We are interested whether there is a half plane bounded by a line through c that contains all the extremal rays—in other words, whether at any time during the sweep, there is a single angular interval greater than π containing no rays. In the following, we first analyze the complexity of the motions of the rays during the sweep and then give an algorithm to answer the question of safeness.

The defining features of T and P of the extremal rays change during the sweep of the trap. We define ray-angle functions ϕ_e^- and ϕ_e^+. These functions give a mapping from the amount of shift of the trap to the angles of extremal rays (see Fig. 3). An edge e need not be supported at all times during the sweep. Hence, these functions are only partially defined. We will argue that the total combinatorial complexity of ϕ_e^- and ϕ_e^+ is $O(m)$. We note that the intersection of the edge e and a feature of T leads to a constant complexity curved part of one of the functions. The total combinatorial complexity of ϕ_e^- and ϕ_e^+ is therefore bounded by the number of features of T, which is $O(m)$.

Now consider a graph of all ray-angle functions of all edges of P in the (x, ϕ)-plane. Here, an x-value corresponds to the amount of shift of the trap. A vertical line in the graph (i.e., a line with a fixed x-value intersects $O(n)$ functions). If the distance between two function values is greater than π for some x, then the part is unsafe at the corresponding position of the trap and hence rejected.

We check this condition using a frequently used geometric technique called a *sweep line algorithm*. We sweep a vertical line ℓ across the graph in the (x, ϕ)-plane. While we do so, we keep track of the ray-angle functions intersecting ℓ. The
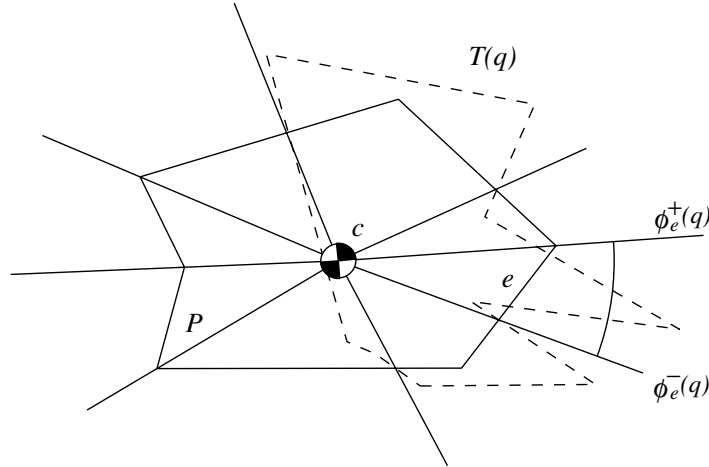
Fig. 3. The part P and its center of mass c above a trap T at placement q. The extremal rays are drawn for all edges of P. For edge e, $\phi_e^-(q)$ and $\phi_e^+(q)$ are shown.

description of the ray-angle functions intersecting the sweep line is called the *status* of the sweep line. For details on sweep line algorithms, we refer to the book of de Berg et al. (1997).

During the sweep, the status needs to be updated at specific values of x. The values at which we update the status of ℓ are called *events*. First, there are events for x-values at which there is an endpoint of a segment of a ray-angle function, since at these values, a ray-angle function must be inserted into or removed from the status structure. Also, there are events when two ray-angle functions change order. Finally, there are events at which two neighboring ray-angle functions have distance π.

When we process an event, we first check which type of event we are dealing with. If the event is due to two neighboring ray-angle functions becoming π apart, we check whether the two functions are indeed still neighboring. In this case, the part is unsafe, and we reject it. An event due to a begin- or endpoint of a ray-angle function segment forces insertion or deletion of a ray-angle function in the status structure. The last kind of event raises the need to update the order of the values of the ray-angle functions in the status structure. From the changes in the status structure, we compute new events.

The reader might have noticed that it can occur that, due to an update of the status structure, the events that correspond to two rays that make an angle of π become invalid. We do not remove them from the set of upcoming events, but we recheck, as mentioned before, the validity of these events at the moment they are processed.

In our case, the status structure is implemented as a balanced binary tree storing the order in which the ray-angle functions are intersected by the sweep line. Since there are $O(n)$ ray-angle functions present in the intersection with the sweep line, the updates and checks take $O(\log n)$ time. The events are stored in a priority queue. For adjacent functions,

we compute their intersections, as well as the x-value for which they are π apart, and enqueue these events.

There are $O(n)$ partially defined ray-angle functions of combinatorial complexity $O(m)$. Each pair of ray-angle functions intersects at most $O(m)$ times. Thus, each pair of ray-angle functions introduces $O(m)$ events. There are $O(n^2)$ pairs of ray-angle functions. Hence, the total number of events is bounded by $O(n^2 m)$.

THEOREM 1.   Let P be a polygonal part with n vertices and T be a polygonal trap with m vertices. We can report whether P is rejected or fed in $O(n^2 m \log n)$ time.

3.2. Convex Traps and Parts

In the case of a convex part and a convex trap, the problem can be solved more efficiently. In this section, we give three lemmas that result in an $O((n + m) \log n)$ algorithm for this case. First, it is shown that the vertices resulting from the intersecting edges of the part and the trap are sufficient to compute the safeness of the part. Lemma 4 shows that we no longer need to consider the supported area of the part outside the trap, but we can confine with $S(q) \cap T(q)$. Second, Lemmas 5 and 6 show that there are only few events, and moreover these events can be processed efficiently, leading to the faster algorithm.

LEMMA 4.   Let P be a convex part with center of mass c at the origin and $T(q)$ be a convex trap at placement q. P is safe if and only if c is in the convex hull of the vertices of $S(q) \cap T(q)$ or $c \notin T(q)$.

Proof. ($\Rightarrow$) Trivial.

($\Leftarrow$) We elaborate on the case that $c \in \mathcal{CH}(S(q)) \cap T(q)$ because if $c \notin T(q)$, the part is evidently safe. We will show

that $\mathcal{CH}(S(q) \cap T(q)) = \mathcal{CH}(S(q)) \cap T(q)$. We prove this by contradiction. Let us assume that there is a point p in $\mathcal{CH}(S(q)) \cap T(q)$ that is not in $\mathcal{CH}(S(q) \cap T(q))$. This point is in $\mathcal{CH}(S(q))$. Consequently, there are two points p' and p'' in $S(q)$, such that p lies on the edge p', p''. Furthermore, p' and p'' have to lie outside $\mathcal{CH}(S(q)) \cap T$ since otherwise p is in $\mathcal{CH}(S(q)) \cap T(q)$ also. But since P is convex, any point on this edge is contained in P, and the intersection points of p', p'' with the boundary of $\mathcal{CH}(S(q)) \cap T(q)$ are evidence for p to lie inside $\mathcal{CH}(S(q)) \cap T(q)$.   □

We restrict ourselves to the part of the motion when $c \in T(q)$, which is a necessary condition for unsafeness of the part. We maintain rays emanating from c, intersecting the vertices of $\mathcal{CH}(S(q) \cap T(q))$, and check whether the angular distance between any pair of neighboring rays remains smaller than π. We shall not explicitly construct the graph of ray-angle functions but rather maintain the ordered set of vertices that are intersected by the rays.

We need events for each q at which the ordered set of vertices of $\mathcal{CH}(S(q) \cap T(q))$ changes combinatorially. The set could change due to appearance or disappearance of vertices from $S(q) \cap T(q)$ or when three vertices of $S(q) \cap T(q)$ become collinear. The following lemma tells us that any event will coincide with a edge-vertex crossing of P and $T(q)$.

LEMMA 5.   The combinatorial structure of $\mathcal{CH}(S(q) \cap T(q))$ only changes when a vertex of $T(q)$ moves across an edge of P, or an edge of $T(q)$ moves across a vertex of P.

Proof.   Suppose that the combinatorial description of $\mathcal{CH}(S(q) \cap T(q))$ changes when there is no vertex-edge crossing. Clearly, no intersection point of P and $T(q)$ appears or disappears. Thus, the combinatorial change is due to the collinearity of the three moving intersection points, v_1, v_2, and v_3. These three points move along three edges of T, e_1, e_2, and e_3. Consequently, there has to be a line intersecting a convex shape through three edges, which is impossible. This completes the proof by contradiction.   □

Hence, there are four possible events where we need to update the combinatorial description of $\mathcal{CH}(S(q) \cap T(q))$ (see Fig. 4).

1. An edge of $T(q)$ moves across a vertex of P, introducing or deleting a vertex of $\mathcal{CH}(S(q) \cap T(q))$.

2. A vertex of $T(q)$ moves across an edge of P, introducing or deleting a vertex of $\mathcal{CH}(S(q) \cap T(q))$.

3. An edge of $T(q)$ moves across a vertex of P, changing the defining edges of a vertex.

4. An edge of $T(q)$ moves across an edge of P, changing the defining edges of a vertex.

LEMMA 6.   The events each require a constant complexity update of $\mathcal{CH}(S(q) \cap T(q))$.

Proof.   The convex hull before the event is denoted by $\mathcal{CH}(S(q) \cap T(q))$ and after the event by $\mathcal{CH}(S(q') \cap T(q'))$.

Case 1. Let v denote the vertex appearing or disappearing from the boundary of $\mathcal{CH}(S(q) \cap T(q))$. Let us assume that v appears on the boundary of the convex hull. The other case is similar. We show that $\mathcal{CH}(S(q) \cap T(q))$ changes locally; that is, an edge (v_1, v_2) changes into two edges (v_1, v) and (v, v_2). Suppose, on the contrary, that some of the vertices from the boundary of $\mathcal{CH}(S(q) \cap T(q))$ do not appear on the boundary of $\mathcal{CH}(S(q') \cap T(q'))$ because they are covered by v (Fig. 5). Let v_a, v, v_b be the neighboring vertices on $\mathcal{CH}(S(q') \cap T(q'))$, after insertion of v. Vertices covered by v are vertices inside the triangle v_a, v, v_b. Let v_c be a covered vertex. Recall that the vertices of $\mathcal{CH}(S(q) \cap T(q))$, as well as the vertices of $\mathcal{CH}(S(q') \cap T(q'))$, are on the boundary of $T(q)$ (respectively, $T(q')$), so v_a, v_b, v_c, and v are on the boundary of a convex polygon. But, v_c lies in the interior of the triangle v_a, v, v_b; therefore, v_c cannot exist, and it follows that the transition from $\mathcal{CH}(S(q) \cap T(q))$ to $\mathcal{CH}(S(q') \cap T(q'))$ is indeed local.

Case 2. A vertex of $\mathcal{CH}(S(q) \cap T(q))$ is split into two vertices, or two vertices merge. This is a local change.

Cases 3 and 4. The edges defining a vertex change. The direction of the motion of the vertex changes, but the trajectory remains continuous. This is a local change as well.   □

By appropriately storing the ordered set of vertices of 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 placements of the trap that the edges of the trap coincide with vertices of the part and vice versa. Therefore, maintaining the convex hull requires $O((n + m) \log n)$ time.

During the motion of the trap, c always has the same distance to the railing. Therefore, at any moment during the motion, there are only two edges of $\mathcal{CH}(S(q) \cap T(q))$ that c 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 placement at which the center of mass 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 placements of the trap at which the center of mass leaves the convex hull. We add these placements as extra events. We handle such events as follows. We first check whether the event is still valid by checking the relevance of the edge associated with the event. If so, we report rejection of the part; otherwise, we discard the event. This gives no extra overhead to the algorithm. The following theorem summarizes the result.

THEOREM 2.   Let P be a convex polygonal part, and let T be a convex polygonal trap. We can report whether P is rejected or fed by T in $O((n + m) \log n)$ time.
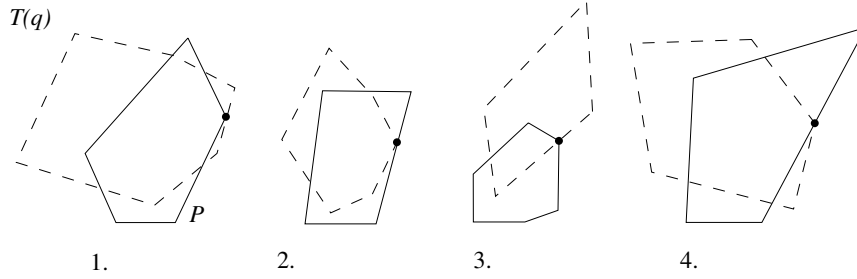
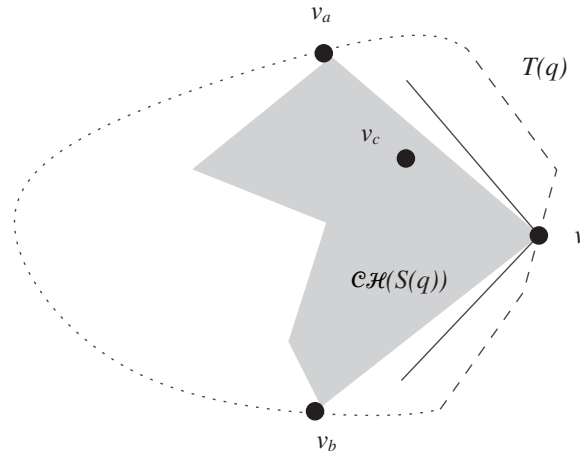Fig. 4. The four possible events types.

Fig. 5. An illustration of Case 1 of the proof of Lemma 6.

4. Design of Traps

In this section, we discuss the design of traps. Given a particular part and a collection of traps (e.g., all rectangular traps), the goal is to find a trap in the collection that satisfies the feeding property (i.e., that allows the part to be fed in only one orientation). We start with various collections of rectilinear traps in Sections 4.1 through 4.4 with increasing numbers of degrees of freedom and conclude with general polygonal traps in Section 4.5.

Figure 6 shows a picture of the rectilinear traps we shall present in the next four subsections—balconies, gaps, canyons, and slots—each with the parameters that define them. The goal of these subsections is to find values for these parameters such that the shape of the resulting trap rejects every orientation of the part, except one.

Clearly, a trap that is entirely contained in another trap will feed all orientations of the latter and possibly more. For a general pair of traps, on the contrary, neither the first trap needs to be contained in the other or vice versa. Consequently, it is hard to order different traps based on the rejection or feedability of traps.

We can, however, for a given orientation σ of P, subdivide the parameter space of all possible trap shapes into shapes that feed P in orientation σ and shapes that reject P in orientation σ.

On the boundaries of the different regions of the subdivision, we find critical trap shapes, which feed the part but have critical placements—only slightly enlarging such a critical trap shape will turn the critical placement into an unsafe placement, turning the trap into a trap that rejects the part. Combining the subdivisions of the trap shapes for different orientations will, on its turn, lead to trap shapes for which only one orientation is fed.

4.1. Balconies

A *balcony* is an interruption of the upper part of the supporting area of the track. The lower boundary, e_l, of this interruption is parallel to the railing. The starting and closing edges of the interruption, e_s and e_c, are orthogonal to the railing. The length of the interruption exceeds the diameter of the part, so that the part cannot simultaneously intersect e_s and e_c. A balcony shape is given by the *balcony width*, μ, which is the distance between e_l and the railing (see Fig. 6a).

We assume that the part is in a fixed stable orientation, so one of its convex hull edges is aligned with the railing. We want to identify a critical balcony. If we start decreasing the balcony width μ from the width of the track to zero, then initially the trap will move across the part without causing the part to fall through. At a certain balcony width, the part
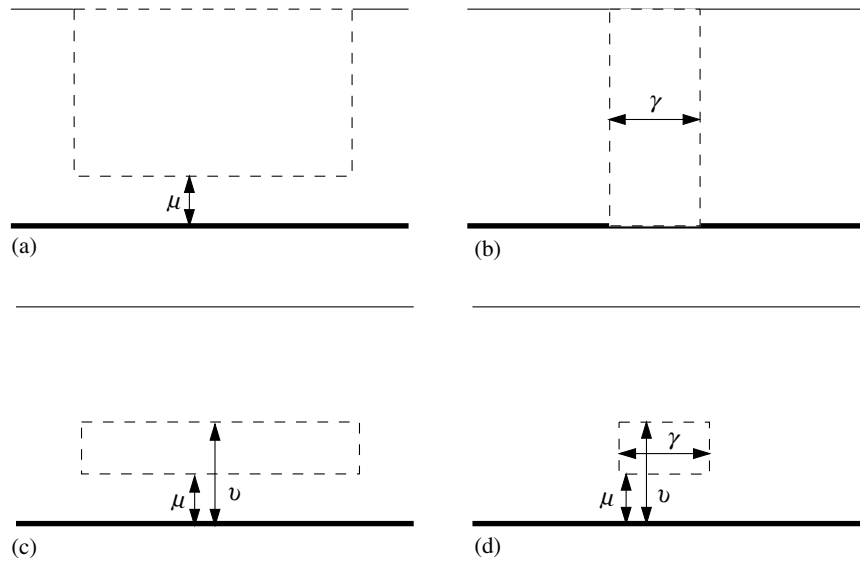
Fig. 6. The four rectilinear traps of this section: (a) a balcony, (b) a gap, (c) a canyon, and (d) a slot. The thick lines at the bottom of the pictures depict the railing. The line at the top depicts the edge of the track at the inside of the bowl. The traps are dashed.

will not survive the balcony, and this clearly remains the case for smaller balcony widths. We refer to the smallest balcony width for which the part survives as the *critical balcony width* for this orientation of the part. If the critical balcony width μ of one stable orientation σ is smaller than the critical balcony widths of all other stable orientations, then a balcony of a width slightly larger than μ (but smaller than all other critical balcony widths) will reject all stable orientations but σ. Hence, this balcony width has the feeding property.

In the following, we show that the critical balcony width for orientation σ corresponds to the distance of the center of mass c to the railing. We denote by H the half plane extending downward from e_l. We observe that $P \cap H \subseteq S(q)$ for any placement q of T. Equality holds for placements for which P is between e_s and e_c. The part in orientation σ is fed by T if and only if P is safe for all placements of T. Since $P \cap H \subseteq S(q)$ for any placement q of T, P is fed if and only if P is safe for placements q of T for which P is between e_s and e_c, as well as $P \cap H = S(q)$. Consequently, by Lemma 1, P is fed if and only if $c \in \mathcal{CH}(P \cap H)$.

A balcony T, for which the width μ equals the distance of c to the railing for P in orientation σ, has c on the boundary of $\mathcal{CH}(P \cap H)$. Thus, placements q of T for which P is between e_s and e_c have c on the boundary of $\mathcal{CH}(S(q))$ and are critical placements, but they still feed P in orientation σ. The distance of c to the railing equals the radius of P in direction σ. Therefore, the critical balcony width for P in orientation σ is radius(σ). In sum, there exists a balcony with the feeding property if the open interval between the two

smallest radii of all stable orientations of P is nonempty (i.e., there is a unique orientation for which the radius is minimal). Since we can compute all radii of P in linear time (Mason 1982), we can determine the balcony widths that have the feeding property in linear time as well.

THEOREM 3. 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.

Proof. The stable orientations and radii corresponding to these orientations of P are computed in linear time. If there is a unique orientation for which the radius is minimal, the feeder is constructed using a balcony slightly higher than this minimum radius. Otherwise, we will always end up with two or more orientations.   □

Note that the railing of the track always touches the part at the convex hull. Therefore, the given analysis holds for both convex and nonconvex parts. The only parts we cannot feed with a balcony are parts for which the minimal radius is not unique. We might also use a balcony at the other side of the track, facing the railing. This "reverse" balcony can be used to select part orientations with a radius greater than the width of the reverse balcony. The combination of a balcony and a reverse balcony in succession on the feeder track is very powerful. Actually, we can select any radius ρ by first rejecting orientations with radii greater than ρ and then rejecting orientations with radii smaller than ρ. So only parts for which each radius occurs more than once cannot be handled in this way.

4.2. Gaps

A *gap* is an interruption of the supporting area that spans the entire width of the track. Both its boundaries are perpendicular to the railing. The shape of a gap can thus be characterized solely by the distance γ between these two parallel boundaries. We shall refer to this distance as the *gap length* (see Fig. 6b).

We again assume that the part is in a fixed stable orientation, so one of its convex hull edges is aligned with the railing. We want to identify a critical gap. If we start increasing the gap length γ from zero to infinity, then initially the trap will move across the part without causing the part to fall through. At a certain gap length, the part will not survive the gap, and this clearly remains to be the case for all larger gap lengths. We refer to the largest gap length for which the part survives as the *critical gap length* for this orientation of the part. If the critical gap length γ of one stable orientation σ is larger than the critical gap lengths of all other stable orientations, then a gap of length slightly smaller than γ (but larger than all other critical gap lengths) will reject all stable orientations but σ. Hence, this gap length has the feeding property. If the largest critical gap length does not correspond to a unique orientation of the part, then there is no gap that can reject all but one orientation of the part, and there exists no gap with the feeding property.

The part is safe if and only if there is a supported triangle around the center of mass. This implies that, when the part is unsafe, 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 unsafe, or critical, placements of the part:

1. The supported area of the part is intersected by at most one edge of the gap.

2. The supported area of the part is intersected by both edges of the gap.

In the first type of unsafe placements, the part is only supported to the left (or the right) of the vertical axis through the center of mass. For the second type of unsafe placements, the supports are contained in a half plane below (or above) the center of mass. The corresponding critical placements also have supports on a line through the center of mass. In Figure 7, four types of unsafe placements are given.

The critical gap length for a stable orientation σ equals the smallest gap length associated with the critical placements of the trap of the two types. The first type of placement does not exist as long as the length of the gap does not exceed the radii of the part in the direction $\sigma - \pi/2$ and $\sigma + \pi/2$. Clearly, if one of these radii is less than the gap length, then the part will fall either forward or backward. Thus, the critical gap length is at most $\min\{\text{radius}(\sigma - \frac{\pi}{2}), \text{radius}(\sigma + \frac{\pi}{2})\}$. It is a bit harder to compute the shortest gap length for which the second type of placement does not occur.

We start by investigating 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). Lemma 3 is the basis for our analysis throughout the rest of this section. We let ρ_l and ρ_r be two rays emanating from the center of mass c, such that $S(q)$ is tangent to the left side of ρ_l and the right side of ρ_r. Figure 8 shows a part at a critical placement of the gap that is supported by two sides of the gap.

The supported area of the part now consists of two regions, one to the left of the gap and one to the right of the gap. The center of mass is in the gap. The widest gap length for which these two regions exist is the gap length that exactly spans the part, supporting the left- and rightmost pieces of the part. Unless c is on the line through the outermost vertices of P, this gap length will not correspond to a critical placement—ρ_l and ρ_r will make an angle unequal to π, and the part will be unsupported.

We conclude that we will have to narrow the gap, until we reach a critical placement (i.e., until the angle between the two rays emanating from c makes an angle of π).

We define a function ϕ_l, which links γ_l to the angle of ρ_l with the positive vertical axis (see Fig. 8). The function ϕ_r is defined similarly. Intuitively, moving the left edge of the gap toward c makes the angle of the left ray with the vertical axis smaller, and moving the rightmost edge of the gap toward c will make the angle of the right ray with the vertical axis smaller. We search for the combination of both motions, for which the rays emanating from c eventually make an angle of π. In the following, we shall validate the intuition and show how to compute the different gap lengths that correspond to critical placements.

An important first observation is that ρ_l will always touch $\mathcal{CH}(S(q))$ at the top. Hence, only the points of P with the maximum y-value for each x-value are important. The union of these points is called the *upper envelope* of the part. The upper envelope can be computed in $O(n \log n)$ time, using an algorithm for computing the upper envelope of line segments by Hershberger (1989).

The second observation is that supported points of P for γ_l are also supported for γ_l' with $\gamma_l' < \gamma_l$. Hence, the supported area to the left of c only increases as γ_l decreases, and consequently, ϕ_l is monotonic.

From the two observations, it follows that a geometric representation of ϕ_l can be computed in two stages. First, we compute the upper envelope of P. Second, we transform the upper envelope into a shape for which the value of $\rho_l(\gamma_l)$ coincides with the intersection of the vertical line at distance γ_l from the vertical axis. We call this shape the *upper tangent envelope*.

The upper tangent envelope for the left side of P can be incrementally constructed by traversing the vertices of the upper hull of P from left to right. We start the traversal at the leftmost vertex v_1 of P. The upper tangent envelope is given by the line segment (v_1, c). As we travel along a
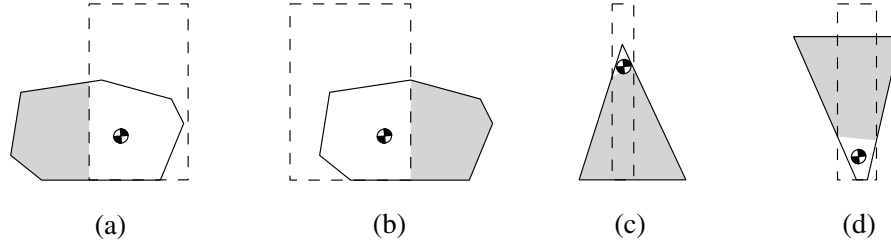
Fig. 7. The types of unsafe placements of a part above a gap. $\mathcal{CH}(S(q))$ is shaded. (a) The supports are to the left of the center of mass. (b) The supports are to the right of the center of mass. (c) The supports are in a half plane below the center of mass. (d) The supports are in a half plane above the center of mass.
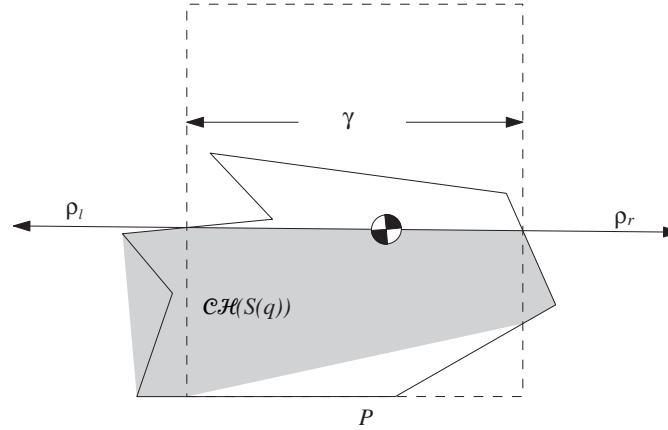

Fig. 8. A critical placement of the gap for the part. ρ_l and ρ_r are rays emanating from c, touching $\mathcal{CH}(S(q))$ on either side.

vertex v_i, there are two possibilities to augment the upper tangent envelope. We consider the line segment (v_{i-1}, v_i). If the segment lies (partially) above the upper tangent envelope computed so far, we add two segments to the upper tangent envelope: the segment of (v_{i-1}, v_i) above the upper tangent envelope and (v_i, c). Otherwise, we discard v_i. We stop at the artificial vertex at the intersection of the vertical axis (through c) and P.

The upper tangent envelope is a representation of ϕ_l and ϕ_r. The value of $\phi_l(\gamma_l)$ is given by the angle of the ray emanating from c to the intersection of the vertical line at distance γ_l from the vertical axis.

The next step is to find all values of γ_l and γ_r for which $|\phi_l(\gamma_l) - \phi_r(\gamma_r)| = \pi$. We start with γ_l at v_1 and find the value of γ_r for which $|\phi_l(\gamma_l) - \phi_r(\gamma_r)| = \pi$. We decrease the value of γ_l while maintaining $|\phi_l(\gamma_l) - \phi_r(\gamma_r)| = \pi$. From the monotonicity of ϕ_l and ϕ_r, it follows that by doing this, γ_r never needs to be decreased. Hence, in a single traversal of the edges of the left side of the upper tangent envelope, we can find corresponding edges of the right side of the tangent envelope for which there are values γ_l and γ_r with critical placements of the trap.

Using elementary trigonometry, we compute for each discovered pair of edges the minimal gap length for which there is a critical placement. Altogether, this takes linear time in the complexity of the upper tangent envelope.

LEMMA 7.   For any orientation of the part, the critical gap length can be computed in $O(n \log n)$ time.

THEOREM 4.   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.

Proof. The stable orientations of P are computed in $O(n)$ time. For each stable orientation, we compute the critical gap length in $O(n)$ time. We determine the largest critical gap length γ. If there is only one orientation that has critical gap length γ, then we design a gap of length slightly smaller than γ (but larger than all other critical gap lengths). This gap feeds only the orientation with critical gap length γ. Otherwise, we conclude that any gap will either reject all orientations, or feed at least two orientations, and we report failure.   □

If the part is convex, the upper tangent envelope of the upper hull of the part is simply the boundary of the part. This allows for a faster computation of the feeder gap length. The critical gap length of a given orientation can now be computed in linear time.

THEOREM 5.   In $O(n^2)$ time, we can design a gap with the feeding property for a convex polygonal part with n vertices or report that no such gap exists.

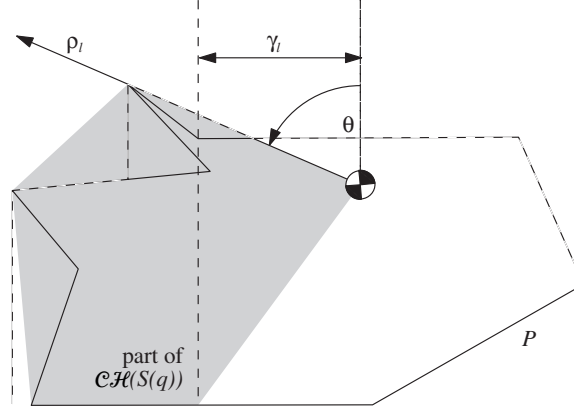Fig. 9. The angle of ρ_l with the vertical axis for a given γ_l. The upper tangent envelope of the part is dashed.

4.3. Canyons

A *canyon* is a rectangular interruption of the supporting area of the track. The lower and upper boundaries, e_l and e_u, of this interruption are parallel to the railing. The starting and closing boundaries, e_s and e_c, of the interruption are orthogonal to the railing. The length of the interruption exceeds the diameter of the part. Hence, there is no placement q of a canyon for which both e_s and e_c intersect $S(q)$ (see Fig. 6c).

We assume that the part is in a fixed stable orientation and seek critical canyons. To this end, we characterize unsafe and critical placements of a canyon. The following lemma allows us to restrict ourselves to placements q of the canyon for which e_s and e_c do not intersect $S(q)$.

LEMMA 8.   Let P be a part. Let T be a canyon. Suppose there is an unsafe placement q of T, with $(e_s \cup e_c) \cap S(q) \neq \varnothing$. There exists an unsafe placement q' of T with $(e_s \cup e_c) \cap S(q') = \varnothing$.

Proof. We suppose without loss of generality that $e_l \cap S(q) \neq \varnothing$. Let $S_u(q)$ denote the area of $S(q)$ above e_u, $S_l(q)$ denote the area of $S(q)$ below e_l, and $S_c(q)$ denote the area of $S(q)$ between e_s and e_c. The canyon is longer than the length diameter of the part, so there exists a placement q' for which P lies between e_s and e_c, and clearly $(e_s \cup e_c) \cap S(q') \neq \varnothing$. $S(q') = (S_u(q) \cup S_l(q)) \subset S(q)$. Hence, any triangle that certifies the safeness of P at placement q' of T also exists in $S(q)$ and consequently certifies the safeness at placement q. This implies, by assumption, that when P is unsafe at q, P is unsafe at placement q' of T as well. $\square$

A consequence of Lemma 8 is that a canyon can be characterized by distances μ and ν, respectively, of the lower and upper boundaries from the railing. Moreover, a critical canyon is characterized by a critical placement q with $(e_s \cup e_c) \cap S(q) = \varnothing$. In the remainder of this section, we focus on placements q with $(e_s \cup e_c) \cap S(q) = \varnothing$. We distinguish two types of unsafe, or critical, placements.

1. The supported area of the part is intersected by at most one edge of the canyon.

2. The supported area of the part is intersected by both edges of the canyon.

In the first type of unsafe placements, the part is only supported above (or below) a horizontal line through the center of mass. For the second type of unsafe placements, the supports are contained in a half plane to the left (or the right) of the center of mass. The corresponding critical placements also have supports on a line through the center of mass.

We denote the height of P in orientation σ by h. Recall that the y-coordinate of c is denoted by c_y. The first type of unsafe placements exists when $\mu = 0$ and $\nu > c_y$, or when $\mu < c_y$ and $\nu > h$. It is a bit harder to compute the second type of critical placements, but they might exist when $0 \leq c_y \leq \nu \leq h$.

We suppose that $S(q)$ lies in a half plane to the right of c (the other case is similar). We derive the dependency between μ and ν in a way that is rather similar to the discussion in Section 4.2.

The supported area of the part, $S(q)$, consists of two regions. One region is above the canyon, and the other is below the canyon. We let ρ_l and ρ_r be two rays emanating from c, such that $S(q)$ is tangent to the left side of ρ_l and the right side of ρ_r. The canyon is at a critical placement if the angle between ρ_l and ρ_r is π. This situation is depicted in Figure 10.

We define two functions, ϕ_l and ϕ_r, which link μ to the angle of ρ_l with the horizontal axis and link ν to the angle of ρ_r with the horizontal axis. We make two observations that will shortly lead to a graphical representation of ϕ_l and ϕ_r. First, we observe that ρ_l and ρ_r always touch $\mathcal{CH}(S(q))$ at one of the leftmost points for any y-value. Second, we observe that the supported area of the part only increases as we increase μ—causing ρ_l to rotate in clockwise direction—or decrease ν—causing ρ_r to rotate in a counterclockwise direction.
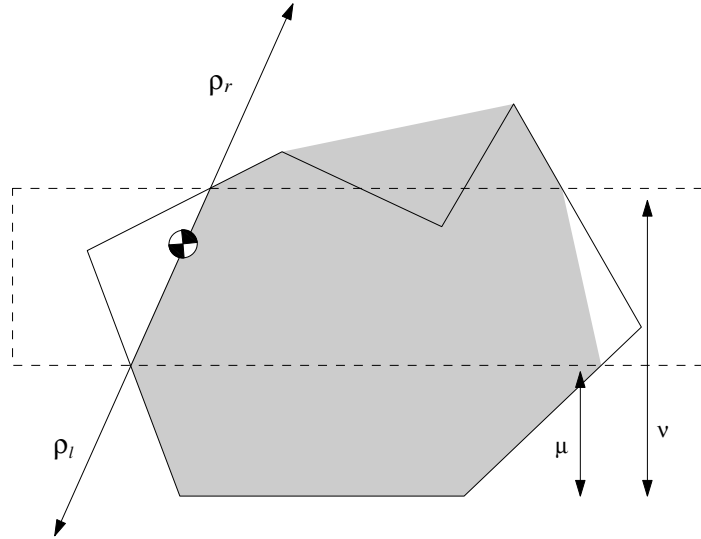
Fig. 10. A critical placement q for the depicted part. ρ_l and ρ_r are rays emanating from c, touching $\mathcal{CH}(S(q))$ on either side.

Hence, ϕ_l and ϕ_r are monotonic, and their representation is given by the left tangent envelope of the part. The left tangent envelope can be computed in $O(n \log n)$ time, similar to the upper tangent envelope of Section 4.2.

The next step is to derive the dependency of ν on μ from the left tangent envelope. We start with $\mu = 0$ and compute ν for which $|\phi_l(\mu) - \phi_r(\nu)| = \pi$. Next, we increase μ while maintaining the collinearity of the rays. From the monotonicity of ϕ_l and ϕ_r, it follows that we never have to decrease ν in this process. We find a linear number of pairs of edges of the left tangent envelope that are simultaneously intersected by the two rays. For every pair of edges, we can compute the dependency between μ and ν, using elementary trigonometry.

We gathered every combination of μ and ν for all critical placements of a canyon for a part in a given orientation. The next step is to combine the combinations for every orientation and select the combinations that only feed one orientation.

To find pairs (μ, ν) that satisfy the feeding property, we draw a graph of all critical canyon shapes (μ, ν) for every orientation of P. The graph consists of $O(n)$ curved segments per stable orientation of the part. The segments are connected, and the relationship between μ and ν is monotonic. We call a connected sequence of these segments a *border*. We draw the border for all possible orientations in the (μ, ν)-plane of all possible canyon shapes.

The border for an orientation σ divides the (μ, ν)-plane into a feeding and a rejecting region for σ. We recall that from the monotonicity of ϕ_l and ϕ_r, it follows that a canyon (μ, ν) that does not have a critical placement for P in orientation σ rejects P in orientation σ if there is a critical placement of a canyon (μ', ν'), with $\mu \geq \mu'$ and $\nu \geq \nu'$. The canyon (μ, ν) feeds P in orientation σ otherwise. Hence, in the graph, the area above the border of σ corresponds to canyons that reject

P in orientation σ and the area below the border to canyons that feed P in orientation σ.

Our ultimate goal is to get a pair (μ, ν) that has the feeding property (i.e., the corresponding canyon feeds only one orientation of the part). A valid pair of (μ, ν) must lie above the border of all but one orientation of the part. Moreover, a valid pair (μ, ν) cannot lie above the borders of all orientations. It follows that it is sufficient to compute the boundary of the set of pairs above the borders of all orientations and the boundary of the set of pairs below the borders of all but one orientation; it is necessary and sufficient for a pair to lie in the area between these two boundaries if it is to have the feeding property.

In the remainder of this section, we will show how to compute the aforementioned boundaries.

In Figure 11, a picture of the border of a single orientation and the intersecting borders of multiple orientations is depicted. The shape that follows the boundary of the set of pairs that lie above the borders of all orientations is the upper envelope of the borders (the upper envelope was introduced in Section 4.2). We compute the upper envelope by means of an algorithm by Hershberger (1989), who computes the upper envelope of a set of segments that intersect pairwise at most k times. The combinatorial complexity of the upper envelope is $\Theta(\lambda_{k+2}(n^2))$, where $\lambda_s(n^2)$ is the maximum length of a Davenport-Schinzel sequence of order s on n^2 symbols (see, e.g., Sharir and Agarwal 1995). The algorithm of Hershberger runs in $O(\lambda_{k+1}(n^2) \log n)$ time. After having computed the upper envelope, we strip the upper envelope from the graph and run the algorithm of Hershberger again to find the boundary of the set of pairs that lie below the borders of all but one orientation of the part. In our case, each pair of segments intersects at most twice.
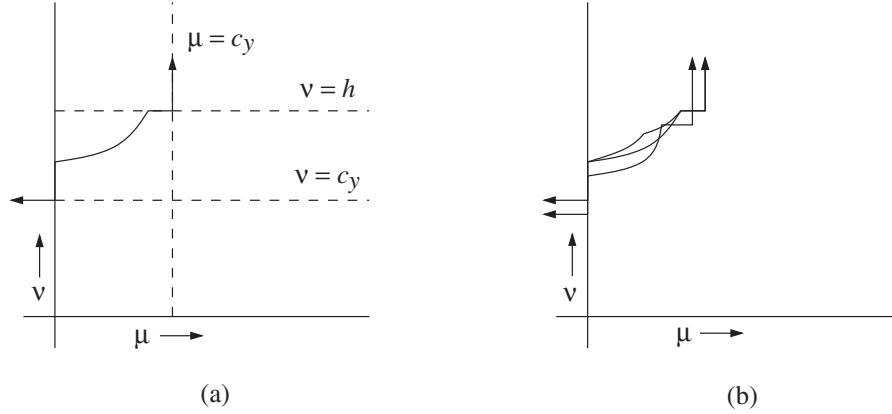
Fig. 11. (a) The border of a single orientation. (b) The graph of borders of multiple orientations.

Since $\lambda_3(n^2) = n^2\alpha(n)$, where $\alpha(n)$ is the extremely slow-growing inverse Ackermann function, we obtain the following theorem.

THEOREM 6.   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.

In the convex case, the running time remains the same since the final step, which computes the upper envelope, dominates the running time of our algorithm.

4.4. Slots

A *slot* is a rectangular interruption of the supporting area of the track. The lower and upper boundaries, e_l and e_u, of this interruption are parallel to the railing. The starting and closing boundaries, e_s and e_c, of the interruption are orthogonal to the railing. The distances of the lower and upper boundaries from the railing are specified by μ and ν, respectively. The length of the interruption is γ (see Fig. 6d).

The strategy to determine critical placements of the part is a combination of the approach of Sections 4.2 and 4.3. We distinguish two types of unsafe, or critical, placements.

1. The supported area of the part is intersected by at most one edge of the slot.

2. The supported area of the part is intersected by more than one edge of the slot.

To characterize the critical placements of a slot, we recall Lemma 3. A slot T is at a critical placement q if two rays emanating from c in the opposite direction are both tangent to $\mathcal{CH}(S(q))$.

Since only two rays determine a critical placement, two edges of the slot are sufficient to determine a critical placement. Consequently, at most, two out of three of the parameters that describe a slot are necessary to describe a critical slot. In other words, each critical slot has at least one parameter that is "free" (i.e., at least one parameter can be varied without affecting the criticality of the slot shape). Figure 12 shows a critical slot shape with free parameter ν; increasing ν still yields a critical slot shape.

A slot shape is given by the triple (μ, ν, γ). We embed the space of all slot shapes in $\mathbb{R}^3$ and generalize the idea of a two-dimensional graph of borders in $\mathbb{R}^2$ from Section 4.3 to surfaces of critical slot shapes in $\mathbb{R}^3$. This collection of surfaces subdivides the space of all slot shapes into regions of feeding and rejecting slots.

The computation of critical slot shapes for P in orientation σ is rather similar to the computation of the critical gap lengths or canyon shapes of the previous sections. For any pair of boundary edges of the slot, we determine the relationship between the angle of collinear rays touching $\mathcal{CH}(S(q))$ and the corresponding slot parameters. This results in a collection of $O(n)$ critical surfaces in the space of slot shapes.

We briefly discuss an algorithm that computes the resulting subdivision of $\mathbb{R}^3$. For each orientation, there are $O(n)$ surfaces of constant algebraic complexity. Thus, the arrangement consists of $O(n^2)$ surfaces in a three-dimensional space. We can compute sample points of the cells in the subdivision, using an algorithm by Basu, Pollack, and Roy (1996). This algorithm also computes at which side of the surfaces the sample points are located, and hence we can determine which orientations are fed and which orientations are rejected for any sample point. Lemma 9 gives the computation time and the number of points computed by their algorithm.

LEMMA 9.   (Basu, Pollack, and Roy 1996).   Let $\mathcal{P} = \{P_1, \ldots, P_\xi\}$, a set of surfaces of constant algebraic degree d in R^l. Then there exists an algorithm that outputs sample points of all semi-algebraically connected components induced by the set $\mathcal{P}$. The complexity of the algorithm is bounded by $\xi^{l+1}d^{O(l)}$.

In our case, $l = 3$ and $\xi = n^2$. Hence, in $O(n^8)$ time, we can compute representatives covering all combinatorially different slot shapes. For each representative, we test whether it has the feeding property. If there does not exist such a
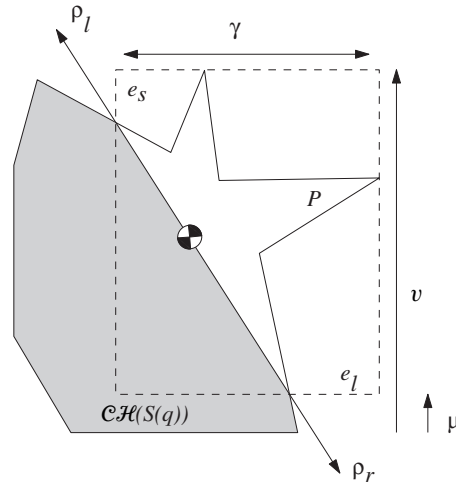
Fig. 12. A critical slot shape for the depicted part. Parameters γ and μ establish a critical placement certified by the rays. Increasing ν still yields a critical slot shape.

representative, there is no slot shape that has the feeding property. The following theorem summarizes the result of this section.

THEOREM 7.   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.

The running time of the algorithm of this section can most likely be improved. The main goal of the presentation of the slot feeder is, however, to give an introduction to computing trap shapes from subdivisions of higher dimensional trap shape spaces. In the next section, we shall compute traps with an arbitrary number of degrees of freedom.

4.5. General Polygonal Traps

In this section, we will show how to design a general trap. Our goal here is not to provide an optimal algorithm but to give a general framework.

The proposed general trap is a polygon with k vertices. The position of each vertex of the polygon is specified by two parameters. This implies that a general polygon can be specified by $2k$ parameters.

The problem of this section is as follows. Let P be a polygonal part and let k be an integer. Design a polygonal trap with k vertices such that P is rejected by the trap in all but one stable orientation as the trap moves across P. Like in the previous sections, we construct a subdivision of the space of possible trap shapes. Since a trap shape is determined by $2k$ parameters, the trap shape space in this section is $\mathbb{R}^{2k}$. The computations that lead to a subdivision of the trap shape space will be carried out in a larger dimensional space. More specif-

ically, the computations will be carried out in an Euclidean space which is spanned by the $2k$ parameters of the trap: the position of the part and the (x, y)-plane.

We compute surfaces that correspond to critical placements of the trap. We follow an approach that is related to robot motion planning, using a cell decomposition. We refer the reader to Latombe (1991) for an overview of robot motion planning and Schwartz and Sharir (1983) on a solution to the general motion planning problem, using an arrangement of higher dimensional, algebraic surfaces.

Our approach to compute the safe placements of the part uses Tarski sets, which are semi-algebraic sets. For ease of presentation, we again describe the problem as if the part will remain stationary during the motion, and the trap moves across the part.

We shall denote the trap, specified by a $2k$-dimensional vector $\tau \in \mathbb{R}^{2k}$, at position $q \in \mathbb{R}$, by $T_\tau(q)$. Let $\Lambda_{\text{int}(T_\tau(q))}(\cdot)$ be the defining formula of the translated trap in $\mathbb{R}^2 \times \mathbb{R}^{2k}$; that is, $\Lambda_{\text{int}(T_\tau(q))}(v)$, $v \in \mathbb{R}^2$ is true if and only if $v \in \text{int}(T_\tau(q))$. Let $\Lambda_P(\cdot)$ be the defining formula of the part in $\mathbb{R}^2$; that is, $\Lambda_P(v)$, $v \in \mathbb{R}^2$ is true if and only if $v \in P$. We assume that $\Lambda_P(\cdot)$ and $\Lambda_{\text{int}(T_\tau(q))}(\cdot)$ are semi-algebraic sets. The intersection of the part and the supporting area of the track, $P - \text{int}(T(q))$, are denoted by the following Tarski set S_1.

$$S_1 = \{(v, \tau, q) \in \mathbb{R}^{2k+3} \mid \Lambda_P(v) \wedge \neg\Lambda_{\text{int}(T_\tau(q))}(v)\}.$$

To determine the safe placement of the part, we take points of S_1 and interpolate between points with the same q and s; that is, we construct the convex hull of the supported area of the part in the direction of the plane for each trap size and trap position. This leads to the set
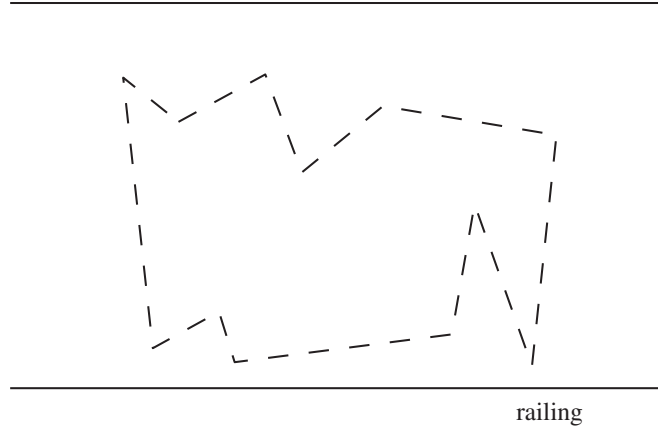
railing

Fig. 13. A general polygonal trap. The position of the vertices is parameterized.

$$S_2 = \{(v, \tau, q) \in \mathbb{R}^{2k+3} \mid \exists v' \in \mathbb{R}^2 \exists v'' \in \mathbb{R}^2 \exists i \in [0, 1] :$$
$$v = iv' + (1-i)v'' \wedge$$
$$(v', \tau, q) \in S_1 \wedge (v'', \tau, q) \in S_1\}.$$

Remember that a placement is safe if its center of mass is inside the convex hull of the supported area of the part. The safe trap shapes in $\mathbb{R}^{2k}$ are the shapes for which there exist no unsafe placement of the trap. A few easy transformations transform the set S_2 into a lower dimensional arrangement that captures the safeness of the part. The only portion of the arrangement that is of interest is the portion that corresponds to the center of mass of P. Therefore, we intersect S_2 with the $(2k+1)$-dimensional space corresponding to the position of the center of mass. This results in

$$S_3 = \{(\tau, q) \in \mathbb{R}^{2k+1} \mid (c, \tau, q) \in S_2\}.$$

If, during the motion, the center of mass is not supported at some placement, the part is rejected. Therefore, we project the complement of the arrangement onto $\mathbb{R}^{2k}$, obtaining

$$S_4 = \{\tau \in \mathbb{R}^{2k} \mid \exists s \in [0, 1] : (\tau, q) \notin S_3\}.$$

We now have a description of a subdivision of the trap shape space $\mathbb{R}^{2k}$ for a single stable orientation into fed and rejected cells. For each orientation of the part, we compute this arrangement. The next step is to merge the m different S_4 arrangements and find a cell for which all but one orientation is rejected. Let us denote S_4 for orientation i by $S_4(i)$. In the remainder of this section, we discuss how to compute a trap that only feeds the first stable orientation of the part and reject all other orientations of the part if such a trap shape exists. Repeating this procedure for the other orientations completes our extensive search for a feeder. Possible trap shapes that feed the first orientation are given by

$$S_5 = \{\tau \in S_4(1) \mid \forall o \in [2, \ldots, m] : \tau \notin S_4(o')\}.$$

Note that o is not a real algebraic variable, and its universal quantifier represents an ordinary for-loop. Unfortunately, the remaining quantifiers found in the expansion of $S_4(i)$ are harder to deal with.

To be able to eliminate these quantifiers, we first transform S_5 into an equivalent sentence with the quantifiers to the left. This is standard procedure and can be found in, for example, the book by Mishra (1993). We denote the resulting formula by S. Traditionally, elimination of the quantifiers in S can be done by Collins' (1975) decomposition, which is a (doubly) exponential algorithm in the number of vertices of the trap. The output of Collins' algorithm are the cells in the arrangement in $\mathbb{R}^{2k}$ of any dimension.

We can improve the running time of the quantifier elimination algorithm by using recent techniques from real algebraic geometry. For a survey, we refer the reader to Heintz, Recio, and Roy (1991); Mishra (1993); and Chazelle (1994). For a comprehensive introductory discussion to the results cited in the following, refer to the thesis of Basu (1996).

We observe several interesting properties of our formula S. Although the number of free variables of S is only bounded by $O(k)$—the number of vertices of the trap—the number of quantified variables is bounded by a constant. Also, the degree d of the polynomials in S is bounded by a constant.

If we would settle for only the semi-algebraic description of the surfaces (without decomposing the space of possible trap shapes into connected components), we could use the recent algorithm of Basu (1999), which benefits from the special properties of our set S and uses singly exponential time in our case. The following lemma states that we can remove the quantifiers from our formula S by computing a set of surfaces

that induces a fine subdivision of the trap shape space, which is equivalent to our formula $\mathcal{S}$.

LEMMA 10. (Basu 1999). Let l and ω be constants and $\mathcal{P} = \{\wp_1, \ldots, \wp_\xi\}$ be a set of ξ polynomials, each of constant degree d, in $l + \kappa$ variables, with coefficients in a real closed field R. And let

$$\Phi(Y) = (Q_\omega X^{[\omega]}) \cdots (Q_1 X^{[1]}) F(\wp_1, \ldots, \wp_\xi),$$

be a first-order formula, where $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_{i+1}$, $Y = (Y_1, \ldots, Y_\kappa)$ is a block of κ free variables, $X^{[i]}$ is a block of a constant number of variables, and $F(\wp_1, \ldots \wp_\xi)$ is a quantifier-free Boolean formula with atomic predicates of the form $\wp_i(Y, X^{[\omega]}, \ldots, X^{[1]}) \{<, >\} 0$. Moreover, let every polynomial in $\mathcal{P}$ depend on at most a constant number of the Y_js. Then there exists an equivalent set of surfaces, $\Psi(Y)$ of size $\xi^{O(1)} d^{O(\kappa)} |F|$, where $|F|$ is the length of the formula F. The algebraic degrees of the surfaces in $\Psi(Y)$ are bounded by a constant.

In our case, ξ and $|F|$ are $O(knm) = O(kn^2)$, and κ is two times the number parameters of the trap. Hence, there is a constant c, such that the output of the algorithm of Basu (1999) is a quantifier-free formula of size $O((kn)^c d^{O(k)})$ and has constant-degree polynomials. The algorithm uses $O((kn)^c d^{O(k)})$ arithmetic operations. The polynomials in the quantifier-free formula represent surfaces of constant degree in the trap shape space. The output of the algorithm is a set of surfaces that subdivides the space of possible trap shapes into regions for which the sets of rejected and fed orientations are fixed.

From Lemma 9, it follows that we can compute sample points in every connected component of the subdivision. We fill in the variables of Lemma 10. From the number of surfaces, it follows $\xi = O((kn)^c d^{O(k)})$. The surfaces are embedded in $\mathbb{R}^{2k}$, and hence $l = 2k$. We conclude that we can compute representatives covering all combinatorially different k-vertex trap shapes using $O((nk)^{O(k^2)})$ arithmetic operations. For each representative, we test whether the resulting trap only feeds the first orientation of P. If there does not exist such a representative, there is no k-vertex polygon that only feeds the first orientation of P. Repeating the algorithm for each stable orientation of P yields the following result.

THEOREM 8. 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.

5. Experimental Results

The trap design algorithms in this paper assume the geometric model presented in Section 2. How well do the resulting traps perform in practice? We discuss some of the differences between theory and practice and then describe laboratory experiments with a physical feeder (Berretty et al. 1999).

Our model of the bowl feeder makes several simplifying assumptions. First, we assume that parts are singulated (i.e., that they do not tangle or nest together, which would create hybrid shapes with varying boundaries and centers of mass). Second, we assume that part motion along the track is frictionless, continuous, and quasi-static. This approximates the true motion in a vibratory bowl, which uses submillimeter helical vibrations to cause parts to advance along the track via a series of microscopic hops. Since these hops are very small with respect to the part dimensions, our assumptions are reasonable. Larger magnitude vibrations could induce discontinuities in the position of the part that could allow it to "hop across" a trap, especially if the trap is designed with tight tolerances. Third, we assume all motion is planar: parts with nonzero vertical extent can become jammed in a trap as they fall. Last, we assume that the outer (vertical) track railing is linear and not curved, as is the case with bowl feeders that have circular railings. If this curvature is small with respect to part dimensions, our linear assumption is reasonable. We encourage future research that can relax some of the assumptions above.

To study the behavior of our traps in practice, we tested two traps in the laboratory (Berretty et al. 1999). Our experimental track, shown in Figure 14, uses a commercial inline vibratory platform from Automation Devices, Inc. The platform generates an asymmetric vibration at variable amplitudes that moves parts along the track. The parts are standard fluorescent lightbulb sockets, approximately 2 inches in length. We assume that parts are singulated and that the same part face lies on the feeder track. Projecting the part onto the track yields a polygonal shape that we provide as input to our trap design algorithms. The two traps were designed along the line of thought discussed at the end of Section 4.1; two traps were combined to enable us to choose the orientation to be fed. The balcony rejects orientations with a radius larger than the fed orientation. The slot was designed to retain the fed orientation and reject the remaining orientations. The balcony and slot output from the algorithms were cut into sheet metal with a milling machine and attached to the vibratory platform.

In a controlled series of 100 trials with each of the part's stable orientations, we experienced no failures: all undesired part orientations were properly rejected by the pair of traps. The traps never jammed; they successfully rejected many out-of-plane part orientations not modeled by the algorithms. Failures were observed in cases when parts were not singulated: pairs of overlapping parts could be arranged to slip past the traps. Also, residual glue from a price label caused one part to violate our motion model.

6. Discussion

In this paper, we have presented a geometric framework for the trap design problem and reported algorithms for the analysis
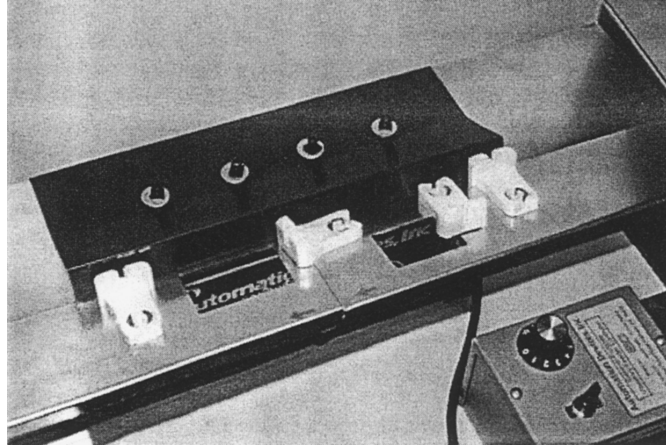
Fig. 14. Part on track and railing mounted on Model 5300A.1 (T-18) vibratory platform from Automation Devices, Inc. Approximate length = 18 inches. The traps were designed by our algorithm and cut with a milling machine. The feeder successfully feeds a stream of these parts.

and design of various traps for polygonal parts moving across a feeder track. We are not aware of any previous geometric algorithms for the systematic design of vibratory bowl feeder traps.

Some of our algorithms have been implemented, and various traps have been shown to work, both in simulation and in practice (Berretty et al. 1999). Many open problems remain. First, optimality of the algorithms has not been proven, and we expect that some of the algorithms can be improved. In particular, it would be interesting to improve the bounds for general polygonal parts. A second question involves the notion of uncertainty in part shape and motion of the part. For a preliminary treatment of this issue, see Berretty (2000).

These results for geometric trap design suggest a variety of new research problems. We want to extend these results to parts with curved edges and to extend the treatment to a full three-dimensional analysis. One approach to three-dimensional parts is to consider each stable orientation of the three-dimensional part as a distinct two-dimensional footprint and then to design a sequence of traps that feeds only one footprint in one orientation. We will also treat out-of-plane effects, in which three-dimensional volumes begin to fall into a trap but become wedged. It would be helpful to know what part geometries cannot be fed using traps and to develop design algorithms that combine traps with other bowl feeder devices such as steps and wiper blades.

Acknowledgments

References

Agarwal, P. K., Collins, A. D., and Harer, J. L. 2001. Minimal trap design. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2243–2248.

Akella, S., Huang, W., Lynch, K. M., and Mason, M. T. 2000. Parts feeding on a conveyor with a one joint robot. *Algorithmica* 26:313–344.

Akella, S., and Mason, M. T. 1992. Posing polygonal objects in the plane by pushing. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2255–2262.

Basu, S. 1996. Algorithms in semi-algebraic geometry. Ph.D. thesis, Department of Computer Science, New York University.

Basu, S. 1999. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM* 46(4):537–555.

Basu, S., Pollack, R., and Roy, M.-F. 1996. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM* 43:1002–1045.

Berkowitz, D., and Canny, J. 1996. Designing parts feeders using dynamic simulation. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1127–1132.

Berretty, R.-P. 2000. Geometric design of part feeders. Ph.D. thesis, Institute of Information and Computing Sciences, Utrecht University, the Netherlands.

Berretty, R.-P., Goldberg, K. Y., Cheung, L., Overmars, M. H., Smith, G., and van der Stappen, A. F. 1999. Trap design for vibratory bowl feeders. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2558–2563.

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

Böhringer, K.-F., Bhatt, V., Donald, B., and Goldberg, K. Y. 2000. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica* 26:389–429.

Boothroyd, G., and Dewhurst, P. 1983. *Design for Assembly: A Designers Handbook*. Amherst: Department of Mechanical Engineering, University of Massachusetts.

Boothroyd, G., Poli, C., and Murch, L. 1982. *Automatic Assembly*. New York: Marcel Dekker.

Brokowski, M., Peshkin, M. A., and Goldberg, K. Y. 1995. Optimal curved fences for part alignment on a belt. *ASME Transactions of Mechanical Design* 117:27–34.

Caine, M. E. 1994. The design of shape interaction using motion constraints. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 366–371.

Chazelle, B. 1994. Computational geometry: A retrospective. *Annual ACM Symposium on Theory of Computing*, pp. 75–94.

Chen, Y.-B., and Ierardi, D. J. 1995. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica* 14:367–397.

Christiansen, A., Edwards, A., and Coello, C. 1996. Automated design of parts feeders using a genetic algorithm. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 846–851.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *GI Conference on Automata Theory and Formal Languages*, LNCS, Vol. 33, pp. 134–183.

de Berg, M., van Kreveld, M., Overmars, M. H., and Schwarzkopf, O. 1997. *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag.

Erdmann, M. A., and Mason, M. T. 1988. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation* 4:367–379.

Goldberg, K. Y. 1993. Orienting polygonal parts without sensors. *Algorithmica* 10(2):201–225.

Heintz, J., Recio, T., and Roy, M.-F. 1991. Algorithms in real algebraic geometry and applications to computational geometry. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 6:137–163.

Hershberger, J. 1989. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters* 33:169–174.

Jakiela, M., and Krishnasamy, J. 1993. Computer simulation of vibratory parts feeding and assembly. *International Conference on Discrete Element Methods*, pp. 403–411.

Latombe, J.-C. 1991. *Robot Motion Planning*. Boston: Kluwer Academic.

Lim, L., Ngoi, B., Lee, S., Lye, S., and Tan, P. 1994. 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.

Lynch, K. M., and Mason, M. T. 1996. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research* 15(6):533–556.

Mason, M. T. 1982. Manipulator grasping and pushing operations. Ph.D. thesis, MIT.

Maul, G., and Thomas, M. 1997. A systems model and simulation of the vibratory bowl feeder. *Journal of Manufacturing Systems* 16(5):309–314.

Mishra, B. 1993. *Algorithmic Algebra*. New York: Springer-Verlag.

Natarajan, B. K. 1989. Some paradigms for the automated design of parts feeders. *International Journal of Robotics Research* 8(6):89–109.

Peshkin, M. A., and Sanderson, A. C. 1988a. The motion of a pushed sliding workpiece. *IEEE Journal of Robotics and Automation* 4(6):569–598.

Peshkin, M. A., and Sanderson, A. C. 1988b. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation* 4(6):696–701.

Riley, F. J. 1983. *Assembly Automation: A Management Handbook*. New York: Industrial Press.

Schwartz, J. T., and Sharir, M. 1983. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics* 4:289–351.

Sharir, M., and Agarwal, P. K. 1995. *Davenport-Schinzel Sequences and Their Geometric Applications*. New York: Cambridge University Press.

Wiegley, J. A., Goldberg, K. Y., Peshkin, M., and Brokowski, M. 1997. A complete algorithm for designing passive fences to orient parts. *Assembly Automation* 17(2):129–136.