Minimal Trap Design

Pankaj K. Agarwal[*], Anne D. Collins[†] and John L. Harer[‡]

Abstract

This paper addresses the issue of trap design for sensorless automated assembly. First, we present a simple algorithm that determines in $O(nm\alpha(nm)\log(nm))$ time whether an n-sided polygonal part will fall through an m-sided polygonal trap. We then introduce the notion of a *minimal* trap for a polygonal part, and develop an $O(n^{4+\varepsilon})$ algorithm that designs a family of minimal feeders built from these traps. The algorithm is complete in the sense that we can always find a feeder, provided that one exists that rejects and supports the appropriate poses of the part with a little tolerance. Finally, we describe how to modify our minimal traps to ensure the part will actually fall through.

1 Introduction

A *feeder* is a mechanism that takes a collection of parts in arbitrary orientations, and returns a stream of identical parts all oriented the same way. One of the most commonly used mechanisms for sensorless part feeding is the *vibratory bowl feeder* which works as follows: a helical track starting at the bottom of a bowl circles the inside wall as it climbs to the top, and the bowl is set to vibrate in such a way that a part lying on the track is forced to move up. (We ignore the actual mechanics of how this happens, and pretend the motion is smooth; see [4].) Along this track is placed a sequence of obstacles (ramps, wiper blades, traps, etc.) designed to allow only parts in the desired orientation to reach the top; the rest are dumped back into the bowl to start again at the bottom. The obstacles we consider are *traps*, collections of strategically placed holes cut into the track through which undesirable orientations of the part will fall. Researchers have used various techniques including genetic algorithms [5], heursitics [7], and geometric algorithms [3] to design trap-based feeders; see [2] for a detailed survey on this topic. In this paper, we adopt the geometric approach to trap-based feeder design that was first developed in [3].

In Section 2, we describe the model of the bowl feeder that we use. In Section 3, we propose a new algorithm to determine whether a trap drops a part. Our algorithm relies on a considerably simpler data structure than the existing $O((nm(n+m))^{1+\varepsilon})$ algorithm [3], reducing the time to $O(nm\alpha(nm)\log(nm))$. In Section 4, we associate to a part a new class of traps that are *minimal* in the sense that any trap that drops the part must contain one of them. This in turn leads to an algorithm for designing a family of feeders in $O(n^{4+\varepsilon})$ time, under a model of computation in which the roots of a fixed degree polynomial can be computed in $O(1)$ time. Each feeder supports one orientation of the part and drops all others; moreover, the family is complete, in the sense that if a feeder exists that really supports one pose and drops the rest, we will find one. Finally, we describe a way to modify each minimal trap to ensure the targeted part falls through, thereby suggesting at least a heuristic approach to designing a family of realistic feeders through which all undesirable poses actually fall.

2 Model

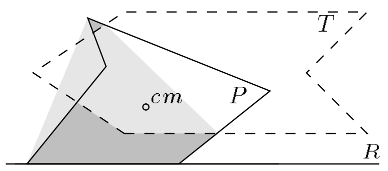
In this section we describe the geometric model of a vibratory bowl feeder. We assume that the bowl is so large relative to the size of the parts that the curvature of the track is negligible. We therefore model the feeder as a straight track along which the parts slide. The track is tilted slightly towards a railing, R, which runs along one side; this ensures that the part remains in contact with R as it moves. We only consider two-dimensional parts, so the entire system can be viewed in the plane.

Let P and T be simple polygons with n and m

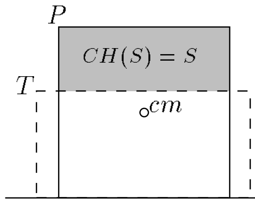
[*]Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129; pankaj@cs.duke.edu. Supported in part by National Science Foundation grants EIA–9870734, EIA–997287, and CCR–9732787, by Army Research Office MURI grant DAAH04–96–1–0013, by a Sloan fellowship, by a National Science Foundation NYI award and matching funds from Xerox Corporation, and by a grant from the U.S.-Israeli Binational Science Foundation.

[†]Department of Mathematics, Box 90320, Duke University, Durham, NC 27708-0320; annec@math.duke.edu. Supported in part by National Science Foundation grant DMS-97-21428

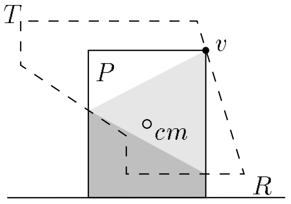
[‡]Department of Mathematics, Box 90320, Duke University, Durham, NC 27708-0320; John.Harer@duke.edu Supported in part by National Science Foundation grant DMS-97-21428



(a) $cm \in CH(S) \Rightarrow P$ is safe.



(b) We assume R does not support, so P falls into the trap here.



(c) $v \in \partial T$ is an isolated point of S. We assume this pose is supported.

Figure 1: Part P (solid) slides along R and overlaps trap T (dashed). The small circle is P's center of mass. The darkly shaded region is S, and the entire shaded region is $CH(S)$.

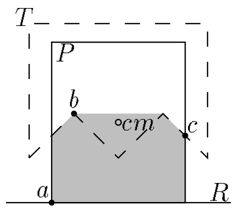


Figure 2: Three types of points contribute to $CH(S)$.

vertices, respectively. P represents the part. The slight tilt in the track not only keeps the part in contact with R as it slides, but also constrains its angular position. As the center of mass seeks to be as low as possible, P will assume one of its $O(n)$ stable orientations; see [6]. The part therefore moves along the track with only one degree of freedom; we assume it moves with unit speed, and parameterize its position by time t.

The trap is a T-shaped hole cut out of the track. It should be noted that, unlike the part, there is no preferred placement of T on the track; thus, its orientation and distance from R must be included in the initial specification of T. We could, for example, assume the coordinates of T are given in a frame where the railing lies along the x-axis. Note that while any x-translation of T yields an equivalent trap, y-translations, rotations and flips do not. We set $t = 0$ to be the time when the centers of mass of P and T have the same x-coordinate.

We also assume that a point of P that lies over an edge of T is supported. We make an exception for those edges of T that lie along the railing; hence the trap is $int(T) \cup int(T \cap R)$.[1] This assumption allows the part in Figure 1(b) to fall through the trap. Abusing notation, T will refer to both the polygon and the trap it defines; it should be clear from context which we mean.

Denote by $P(t)$ and $cm(t)$ the part and its center of mass at time t. Define $S(t)$, the *supported region* of P at t, to be the portion of P that does not lie over the trap at time t, $S(t) = P(t) - (int(T) \cup int(T \cap R))$; see Figure 1.

Define the part to be *safe* or *supported* at t if $cm(t) \in CH(S(t))$; otherwise, $P(t)$ is *rejected* or *dropped* at t. We say T drops P if $P(t)$ is dropped by T for some time t, and P is supported by T otherwise. Note that 'T drops P' does not necessarily mean that the part actually falls through the trap; we defer this issue to Section 4.3.

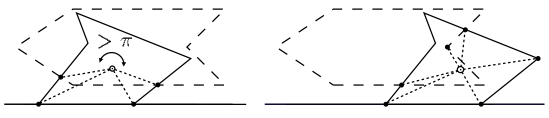
Let $V(t)$, the *support vertices* of $P(t)$ with respect to T, be the following subset of the vertices of $S(t)$: (a) vertices of $P(t)$ outside $int(T) \cup int(T \cap R)$, ($b$) reflex vertices of T in $P(t)$, and (c) intersections of an edge of $P(t)$ with an edge of T (Figure 2). Then $CH(S(t)) = CH(V(t))$. We only need consider reflex vertices of T since a convex vertex creates a reflex vertex of $S(t)$ and thus cannot contribute to $CH(S(t))$. We call vertices of type (c) *mixed* vertices, and include only the endpoints of an improper intersection. The support vertices vary continuously with t as P moves along the track, except at those times when a vertex in $V(t)$ disappears or a new one appears. These events occur when a vertex of $P(t)$ crosses the boundary of T or a reflex vertex of T crosses ∂P.

3 Decision Algorithm

In this section, we describe an algorithm for deciding whether a trap T drops an oriented part P. We also describe how to determine the set of *all* times when P drops, $\{t \mid cm(t) \notin CH(S(t))\}$.

The algorithm of Beretty et.al. [3] maintains $CH(S(t))$ explicitly, using an adaptation of the kinetic convex hull algorithm of [1]. However, only

[1] Whether this assumption is reasonable depends upon the actual design of the bowl feeder. If we were to assume that R supports P, the trap would be $int(T)$ and the part in Figure 1(b) would not fall. It is not difficult to modify our analysis for this situation.



(a) P dropped (b) P supported

Figure 3: The safeness of P depends upon the size of the largest empty cone about cm.

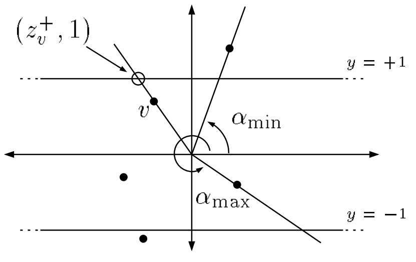


Figure 4: $\alpha_{\min}$, $\alpha_{\max}$ and z_v^+

some of this information is necessary; by maintaining a considerably simpler structure, we achieve a faster algorithm.

Our algorithm is based upon the following simple idea: T drops $P(t)$ if and only if the largest cone centered at cm that is empty of support vertices has angular measure greater than π; see Figure 3. If such an empty cone exists, it must contain the ray from $cm(t)$ in either the positive or negative x direction.

We present two algorithms that determine if and when the angular measure of the largest empty cone that contains the positive x-axis is greater than π. Analogous arguments can be made with respect to the negative x-axis.

3.1 Outline of Algorithm

To simplify our picture, we *change coordinate systems* to one centered at cm. P is now stationary and the trap $T(t)$ moves to the left. Set $t = 0$ when the center of the trap is at $x = 0$.

For each vertex $v \in V(t)$, let $\alpha_v(t) \in [0, 2\pi)$ be its angular position about the origin (cm), and let $\alpha_{\min}$ and $\alpha_{\max}$ be the lower and upper envelopes, respectively, of $\{\alpha_v(t) \mid v \in V(t)\}$ (Figure 4). Then $P(t)$ drops if $\alpha_{\max}(t) - \alpha_{\min}(t) < \pi$.

Let $t_1 < \ldots < t_k$ be the set of all breakpoints of $\alpha_{\min}$ and $\alpha_{\max}$, set $t_0 = -\infty$, $t_{k+1} = +\infty$, and let ν_i and χ_i be the vertices of V that realize each envelope, respectively, in the interval (t_i, t_{i+1}). In order to decide that T drops P, we need only find one t where $P(t)$ drops; thus our decision algorithm proceeds as follows:

Compute t_i, ν_i and χ_i, and check whether $\alpha_{\chi_i}(t_i) - \alpha_{\nu_i}(t_i) < \pi$ for any t_i. If not, check for discrete times $t \in (t_i, t_{i+1})$ when $\alpha_{\chi_i}(t) - \alpha_{\nu_i}(t) = \pi$. If necessary, repeat these tests with respect to the negative x-axis, and report that P is supported if no such time is found.

We need only look for discrete intersections of $\alpha_{\max}$ and $\alpha_{\min} + \pi$, since otherwise $\alpha_{\chi_i}(t) - \alpha_{\nu_i}(t) = \pi$ and the part is supported for all $t \in (t_i, t_{i+1})$. The correctness of the algorithm is otherwise straightforward.

It is also useful to know not only if, but *when* T drops $P(t)$. Let $s_1 < \ldots < s_j$ include $\{t_0, \ldots, t_{k+1}\}$ as well as the proper intersections of $\alpha_{\max}$ and $\alpha_{\min} + \pi$. We show in Section 3.2 that if $\alpha_{\chi_i}(t) - \alpha_{\nu_i}(t) \not\equiv \pi$ then there are at most two solutions, so $j = O(k)$. The following algorithm determines $I = \{t \mid cm \notin CH(S(t))\}$: Initialize $I = \emptyset$, and compute s_i, ν_i and χ_i. Scan the breakpoints and intervals in sequence, adding $\{s_i\}$ to I if $\alpha_{\chi_i}(s_i) - \alpha_{\nu_i}(s_i) < \pi$, and adding (s_i, s_{i+1}) to I if $\alpha_{\chi_i}(\frac{s_i + s_{i+1}}{2}) - \alpha_{\nu_i}(\frac{s_i + s_{i+1}}{2}) < \pi$. We then repeat this with respect to the negative x axis, to obtain a list of all times when P drops.

3.2 Implementation Details

Computing $\{\alpha_v\}$ explicitly involves inverse trigonometric functions, so we use an alternative parametrization which is algebraic.

First, partition $V(t)$ into $V_+(t) = \{v \in V(t) \mid 0 \leq \alpha_v(t) < \pi\}$ and $V_-(t) = \{v \in V(t) \mid \pi \leq \alpha_v(t) < 2\pi\}$. If either $V_+(t)$ or $V_-(t)$ is empty, $P(t)$ drops.

For each $v \in V_+(t)$, project v through the origin to the line $y = 1$ and let $z_v^+(t)$ be the x-coordinate of this point. That is, $z_v^+(t) = \frac{x(v(t))}{y(v(t))} = \cot(\alpha_v(t))$; see Figure 4. Let $R_+(t)$ be the upper envelope of $\{z_v^+(t) \mid v \in V_+(t)\}$; then $R_+(t) = \cot(\alpha_{\min}(t))$ if $0 \leq \alpha_{\min} < \pi$; otherwise, $R_+(t) = -\infty$.

Similarly, project $v \in V_-(t)$ to the line $y = -1$, and let $z_v^-(t) = \frac{x(v(t))}{y(v(t))} = \cot(\alpha_v(t))$ (the *negative* of the x-coordinate of the projected point). Let $R_-(t)$ again correspond to the rightmost of these projected points, in this case the lower envelope of $\{z_v^-(t) \mid v \in V_-(t)\}$. Then $R_-(t) = \cot(\alpha_{\max}(t))$ if $\pi \leq \alpha_{\max} < 2\pi$ and $R_-(t) = +\infty$ if $V_-(t) = \emptyset$.

Lemma 3.1 *$\alpha_{\max} - \alpha_{\min} < \pi \iff R_+ - R_- < 0$.*

If we replace $\{s_i\}$ in the algorithms of Section 3.1 with the breakpoints and proper intersections of R_+ and R_-, and test instead for $R_+(t) - R_-(t) < 0$ we will correctly report if and when $P(t)$ drops.

We have an arrangement of $O(nm)$ arcs, any two of which intersect at most twice. It can be shown [8, Thm 6.5] that the breakpoints of the upper and lower envelopes can be computed in $O(nm\,\alpha(nm)\log(nm))$ time. We therefore conclude the following

Theorem 3.2 *Let P be an oriented polygonal part and T a polygonal trap with n and m vertices, respectively. We can determine whether T drops P in $O(nm\,\alpha(nm)\log(nm))$ time. Moreover, the same bound holds for finding $\{t \mid T \text{ drops } P(t)\}$.*

Remark 3.3 (i) We can reduce the running time in some cases. If P and T can be decomposed into k_P and k_T y-monotone pieces, respectively, it can be shown that there are in fact only $N = O(k_T n + k_P m)$ support vertex curve segments.
(ii) While an edge may define many support vertices at a given time, only the two outer vertices can contribute $CH(S)$. By maintaining only these, we can perform the computation using $O(n+m)$ space.

4 Minimal Trap Design

In this section we develop an algorithm for designing a feeder. More precisely: given oriented polygons P_0, P_1, ..., P_k, we design a family of traps which support P_0 and drop all other P_i. In practice, $\{P_i\}$ may be the stable poses of a single polygon P, and P_0 is the one orientation we want to support.

Assume we have a collection $\{T_i\}_{i=1}^k$ of traps, where each T_i drops P_i and supports P_0. Since we can of course place them anywhere along the track, we now allow a 'trap' to have multiple connected components, provided that each is itself simply connected. Let $T(\tau)$ denote the trap T shifted horizontally by τ. Then the combined trap $\bigcup_{i=1}^k T_i(\tau_i)$ drops each P_i for any choice of $\{\tau_i\}$.

However, not all of these traps work as feeders; it is possible that two of the T_i overlap and drop P_0 (Figure 5). The easiest solution, and the one which we adopt, is to choose $\{\tau_i\}$ so the $T_i(\tau_i)$ are disjoint. The resulting combined trap will drop $\{P_1, \ldots, P_k\}$ and support P_0 as desired.

The remainder of this section is devoted to designing the collection $\{T_i\}$. We introduce our *minimal* traps in Section 4.1, and Section 4.2 outlines an algorithm which selects a subset of these to be used in our feeders. Finally, in Section 4.3, we address whether a part *really* falls and modify our minimal traps to ensure this.

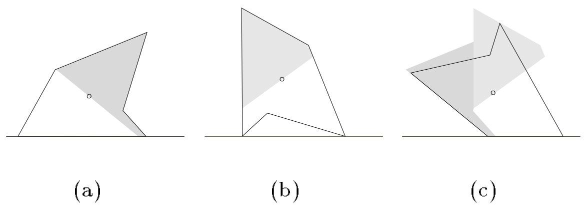


Figure 5: The relative placement of traps is important: alone, the minimal traps in (a) and (b) support the orientation in (c), but their overlap drops it.

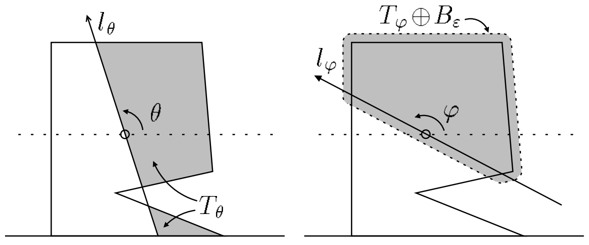
4.1 Minimal Traps

Let P be a simple polygon in one of its stable orientations. For each $\theta \in S^1$, let l_θ be the oriented line in the θ direction that passes through $cm(0)$. Let h_θ be the closed half plane to the right of l_θ and define $T_\theta = P(0) \cap h_\theta$ (Figure 6). Note that l_θ and $l_{\theta+\pi}$ are different; in fact, $T_\theta \cup T_{\theta+\pi} = P(0)$. The trap defined by T_θ, $int(T_\theta) \cup int(T_\theta \cap R)$, is called a *minimal trap* for P. They are minimal in the sense that any trap that drops P must contain T_θ for some θ. As above, we let T_θ refer to both the polygon and the trap it defines.

Unfortunately, T_θ does not drop P at $t=0$; however, any ε-neighborhood of T does (Figure 6(b)). That is, $T_\theta \oplus B_\varepsilon$ drops $P(0)$ for any $\varepsilon > 0$ and any θ, where $\oplus$ is the Minkowski sum, and B_ε is a disk of radius ε. Of course, we really mean $(T_\theta \oplus B_\varepsilon) \cap track$; this implies that if R supports P, not all θ will work.

The real issue here is the presence of the boundary of T; ∂P leads to similar borderline cases. A pose which is supported at t but would drop if ∂T and ∂P were removed is not desirable, as slight fluctuations in the shape of the part or trap make the safeness of P unpredictable. We say P is *almost dropped* at t in this case, and the remaining supported poses are *really supported.* It is not difficult to show that T drops or almost drops $P(t)$ if and only if $cm \notin int(CH(V(t) - V_I(t)))$, where $V_I(t)$ are the isolated vertices of $V(t)$. Furthermore, if T really supports P, there exists an $\varepsilon_0 > 0$ such that $T \oplus B_\varepsilon$ really supports P for all $0 < \varepsilon < \varepsilon_0$; that is, we can make the trap slightly larger without dropping P.

To design a feeder, we choose one minimal trap for each P_i and expand it by ε_i. By insisting that the minimal trap we choose *really* supports P_0 and that ε_i is sufficiently small, we can be sure that this expanded trap supports P_0 as well.



(a) Minimal trap T_θ may be disconnected.

(b) An ε-nbhd of T_φ (dotted) drops P.

Figure 6: Part P and some of its minimal traps.

4.2 Choosing a Minimal Trap

Our goal in this section is to determine which of the minimal traps for P_1 really support P_0. Let T_θ represent the minimal traps with respect to P_1. For fixed θ, the algorithm of Section 3 will answer 'does T_θ support P_0 for all t?', but this is not quite what we want. We need to know 'for *which* θ does T_θ *really* support P_0 for all t?' We now describe how to modify the previous algorithm to address this more difficult question.

The idea remains the same: we maintain the angular measure of the largest empty cones containing the positive and negative x-axes, by partitioning the support vertices about the x axis and defining $z_v^\pm$ and $R_\pm$ as before.

There are, however, two significant differences. First, we test P_0 against T_θ allowing θ to vary. Our support vertices now depend upon θ as well as t; denote by $V(t,\theta)$ the support vertices of $P_0(t)$ with respect to T_θ. Therefore $z_v^\pm(t,\theta)$ are now bi-variate functions and $R_\pm(t,\theta)$ are the upper and lower envelopes of a collection of surface patches.

Second, we want only those θ for which T_θ really supports P_0. That is, we want to determine $\Theta = \{\theta \in S^1 \mid cm(t) \in int(CH(V(t,\theta) - V_I(t,\theta)))\}$, where cm_0 is the center of mass of P_0 and V_I are the isolated vertices of $S(t,\theta)$. We remove the isolated vertices by leaving open some of the boundary edges of the $z_v^\pm(t,\theta)$, and omit the boundary of the convex hull by grouping poses where $R_+ - R_- = 0$ with the 'dropped' poses.

Define $\mathcal{D}_R = \{(t,\theta) \mid R_+(t,\theta) - R_-(t,\theta) \leq 0\}$, and similarly define $L_\pm$ and $\mathcal{D}_L$ with respect to the negative x-axis. Then $\theta \in \Theta$ if and only if the line $\mathbb{R} \times \{\theta\}$ does not intersect $\mathcal{D} = \mathcal{D}_R \cup \mathcal{D}_L = \{(t,\theta) \mid cm \notin int(CH(V - V_I))\}$. Each of these pieces can be determined by computing an appropriate envelope: $\mathcal{D}_R$ is the projection to the t-θ plane of the portion of the lower envelope of $\{R_+, R_-\}$ that is defined by R_+, and similarly for $\mathcal{D}_L$. Θ is the set of all θ for which the left envelope of the boundary curves of $\mathcal{D}$ is $+\infty$.

Assuming P_0 and P_1 both have $O(n)$ vertices, we have an arrangement of $O(n^2)$ surface patches which are algebraic in t, $\cot\alpha$ and $\tan\theta$. It can be shown [8, Ch. 7] that the upper and lower envelopes can be computed in $O\left(n^{4+\varepsilon}\right)$ time. There are $O(n)$ stable orientations, but for any fixed (t,θ) pair we need only $O(n)$ supporting vertices, therefore we can prove that the total running time is $O(n^{4+\varepsilon})$.

Theorem 4.1 *A complete family of feeders which drop $\{P_i\}_{i=1}^k$ and really support P_0 can be determined in $O\left(n^{4+\varepsilon}\right)$ time.*

Here by *complete* we mean that if there exists a feeder which really supports P_0 and drops all other poses, we will find one.

4.3 On Falling

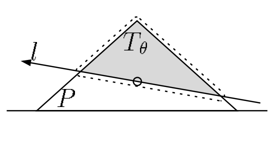
We mentioned early on that 'T drops P' does not guarantee that the part actually falls through the trap. If we want to design a functional feeder, we need to modify our traps in order to ensure that the unwanted poses really fall through.

We first need to make some assumptions about how a part falls into a trap. Let P be any oriented polygon, T a trap which drops it, and v the nearest point of $\partial CH(S)$ to cm. If unsupported, cm will follow the shortest possible downward path. This causes the part to pivot about the line through v orthogonal to the segment $\overline{v\,cm}$. If $cm \in \partial CH(S)$, this line is the edge that contains cm. We call the first such line the *drop line* for P with respect to T.

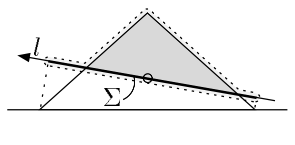
After it starts to tip, we assume the part rotates about the drop line until it is vertical, then slides straight down, if it can (this is certainly an idealized version of what happens).

There are two problems which may arise. First, the part may be too wide. If it is too wide on both sides (Figure 7(a)), the part will not slide through. If only one side is too wide (Figure 6(b)), the part may still slide through, but only after rotating in its vertical plane. We avoid such a complicated motion when possible.

A second problem occurs if the trap has more than one connected component; thus it is a particular problem with minimal traps. If the part is to fall into more than one component of the trap, (Figure 6(a)), it may get stuck straddling the portion of the track between them.



(a) P is too wide to fit through $T_\theta \oplus B_\varepsilon$,



(b) but it does fall through $(T_\theta \cup \Sigma) \oplus B_\varepsilon$.

Figure 7: Σ, the bold portion of l, allows P to fall.

The following patch eliminates both problems. Since we are in particular concerned with fixing minimal traps, we will focus on these; but a similar construct works for any P and T.

Given P, we describe how to augment its minimal traps to ensure that P really falls: Let T_θ be a minimal trap for P, and choose ε so that $T_\theta \oplus B_\varepsilon$ supports P. Let l be the drop line for P with respect to $T_\theta \oplus B_\varepsilon$, let h be the half plane bounded by l that contains S_ε, and let Σ be the convex hull of the orthogonal projection of $P \cap h$ onto l (Figures 7(b), 8).

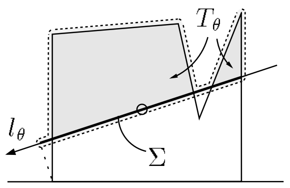
In most situations, P will fall through $(T_\theta \cup \Sigma) \oplus B_\varepsilon$. There are, however, two serious complications: First, Σ may intersect the railing. Since we cannot widen the trap below R, we cannot avoid the part rotating in its vertical plane once it has tipped upright.[2] We assume that the bowl is constructed so that cm cannot lie directly below R (this is reasonable if, say, the wall of the bowl is the railing). Then by rotating in its drop plane, the part will slide through provided the strip along the drop line is long enough. We therefore define Σ in this case to be the segment of the drop line starting at R whose length is the diameter of the part.

Second, $T_\theta \cup \Sigma$ may not be simply connected (Figure 8(b)). This is not a realistic trap as it implies a free-floating portion of the track. We therefore include in the trap the bounded components of $Track - (T_\theta \cup \Sigma) \oplus B_\varepsilon$, which we denote by F.

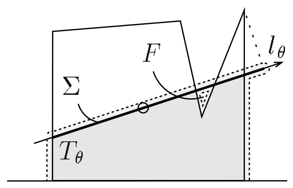
We therefore define our modified minimal traps to be $\hat{T}_\theta(\varepsilon) = ((T_\theta \cup \Sigma) \oplus B_\varepsilon) \cup F$, where Σ is shifted and lengthened at the railing if necessary.

Consider again the feeder design problem, and let $\hat{T}_\theta$ be the modified traps with respect to P_1. We need to determine $\hat{\Theta} = \{\theta \mid \hat{T}_\theta \text{ really supports } P_0\}$. The addition of Σ does not change the safeness of P_0; however, removing larger chunks from the track can make a difference, and $\hat{\Theta}$ may be smaller than

[2] This is the one case whose treatment depends completely upon the design of the actual bowl feeder. We suggest and treat only one possibility.



(a)



(b)

Figure 8: The addition of Σ may (a) join disconnected pieces of T_θ or (b) create a hole (F) in the trap.

Θ. Our algorithm can be modified to compute $\hat{\Theta}$ in $O(n^{4+\varepsilon})$ time; we omit the details.

If the parts $\{P_1, \ldots, P_k\}$ are convex, Σ cannot introduce holes in $\hat{T}_\theta$, and we have a complete family of realistic feeders: $\cup_i \hat{T}_{\theta_i}(\tau_i)$ for any $\{\tau_i\}$ such that the $\hat{T}$ are disjoint. Unfortunately, this does not extend to non-convex parts. A modified minimal trap $\hat{T}_\theta$ for P_1 may drop P_2, but modifying $\hat{T}_\theta$ so that P_2 falls may introduce a hole, thus altering $\hat{\Theta}$ or even the drop line for P_1. While the solution is no longer complete, avoiding angles of $\hat{\Theta}_i$ for which P_j drop still provides a useful heuristic for designing a realistic feeder for non-convex parts.

References

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

[2] R.-P. Berretty. *Geometric Design of Part Feeders.* PhD thesis, Utrecht University, Utrecht, The Netherlands, 2000.

[3] R.-P. Berretty, K. Goldberg, M. H. Overmars, and A. F. van der Stappen. Geometric algorithms for trap design. *Symp. Comp. Geom.*, pages 95–104, 1999.

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

[5] A. Christiansen, V. Chandry, and M.M. Barash. Automatic design of parts feeders using a genetic algorithm. In *Proc. IEEE ICRA*, pages 846–851, 1996.

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

[7] L. Lim, B. Ngoi, S. Lee, S. Lye, and P. Tan. A computer-aided framework for the selection and sequencing of obtaining devices for the vibratory bowl feeder. *Intl. J. Production Research*, 32, 1994.

[8] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, 1995.