2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

On the Design of Traps for Feeding 3D Parts on Vibratory Tracks

Onno C. Goemans and A. Frank van der Stappen *

Abstract— **In the context of automated feeding (orienting) of industrial parts, we study the algorithmic design of traps in the bowl feeder track that filter out all but one orientation of a given polyhedral part. We propose a new class of traps that removes a V-shaped portion of the track. The proposed work advances the state-of-the-art in algorithmic trap design by extending earlier work [1], [3], [11]—which focuses solely on 2D parts—to 3D parts, and by incorporating a more realistic part motion model in the design algorithm. The presented complete design algorithm takes as input any polyhedral part, along with its center of mass, and reports all valid trap designs that feed the given part.**

I. INTRODUCTION

A *part feeder* takes in a stream of identical parts in arbitrary orientations and outputs them in a single orientation. We consider the problem of *sensorless orientation* of parts in which parts are positioned and oriented using passive mechanical compliance.

The oldest and still most common approach to automated feeding is the vibratory bowl feeder. It consists of a bowl, which is partially filled with (identical) parts, and a helical metal track that starts at the bottom and winds its way up along the bowl [5]. The bowl and track undergo an asymmetric helical vibration that causes the 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. A stream of oriented parts emerges at the top after successfully running the gauntlet. A device or sequence of devices is said to *feed* a part if it allows only one specific orientation of the part to pass.

For the manufacturing of feeders, engineers still rely mostly on skill, experience, and ad-hoc guidelines. Currently, the largest cost factor of bowl feeder production is the design of the custom mechanisms tailored to feed a specific part. The expenses and time associated with the design of part feeders remains a barrier to flexible automation.

To aid in the design of bowl feeder layouts, researchers have used simulation [2], [14], [18], [19], [20], heuristics [15] and genetic algorithms [8]. Genetic algorithms have also been used for a programmable bowl setup using beam sensors [21]. Geometric analysis tools have been developed that help designers visualize the configuration space of a given combination of part and bowl layout [7]. Research to single reorientation and rejection mechanisms has been focused at certain classes of simple parts [5], [6] and, for a more general class of parts, at the design of traps [1], [3], [11]—traps are devices constructed by removing sections of the track. A common characteristic of the general solutions in the latter category is that the resulting (trap) designs only apply to two-dimensional parts. A recent piece of work not sharing this characteristic proposes a blade mechanism for feeding polyhedral parts [10].

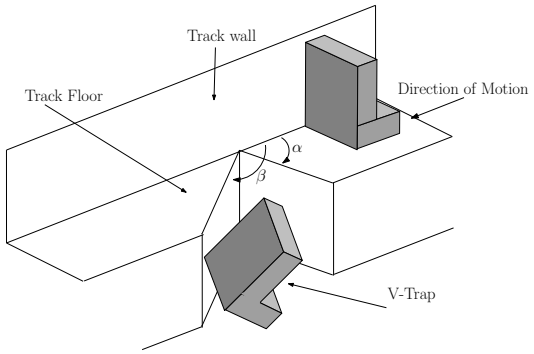

Fig. 1. Vibratory track with V-trap $T(\alpha, \beta)$.

Even in the broader context of sensorless feeding in general, the majority of the achievements apply to flat, two-dimensional parts, or to 3D parts of which the resting face is fixed beforehand [17]. In addition to the aforementioned blade mechanism, three other contributions that also focus on polyhedral parts are work by Berretty *et al.* [4], which focuses on tilted plates with fences, work by Zhang and Gupta [22], which uses step devices, and work by Lynch [16], which employs a robot arm over a conveyor belt.

In this paper, we narrow the gap between the industrial application of vibratory bowls and scientific work on the automated design of sensorless feeders by studying the algorithmic design of traps for feeding 3D parts. Extending prior work on traps [1], [3], [11] for 2D parts, we propose a class of traps for feeding 3D parts that is inspired by similar devices from the practice of vibratory feeder design [5], [6]. We believe that this class of what we refer to as *V-traps* is rich enough to feed a broad class of 3D parts yet simple and structured enough to facilitate efficient automated design.

V-traps are created by removing a V-shaped section of the feeder track between two lines fanning out from the track wall. We characterize a V-trap by two parameters, α and β, which are the angles between the left and right trap edges, respectively, and the track wall (Fig. 1). A trap can thus be described as $T(\alpha, \beta)$. V-shaped traps provide a way to positively influence the deflection of rejected parts

* Onno C. Goemans and A. Frank van der Stappen are with the Dept. of Information and Computing Sciences, Utrecht University, PO Box 80089, 3508 TB Utrecht, The Netherlands. *Email*: onno@cs.uu.nl, frankst@cs.uu.nl. Onno C. Goemans acknowledges the support of the Netherlands Organisation for Scientific Research (NWO). A. Frank van der Stappen is partially supported by the Dutch BSIK/BRICKS-project.

off the track: traps with a larger difference between α and β are preferable [5], [6]. The availability of all solutions, as output by our complete design algorithm, allows for the selection of trap designs with good deflection abilities as a post-processing phase. Except for the one-parameter balcony trap [3], we surmise that the class of V-traps generally provides stronger deflection capabilities than previously proposed classes of traps [1], [3], [11]; further research is needed to confirm this intuition.

In addition to presenting the first design algorithm for traps for feeding 3D parts, we advance the state of the art by incorporating a more realistic model for part motion into the design process. Prior work on the design of rejection devices for vibratory bowls assumes that parts slide smoothly along the track, while in reality they "hop" along the track. In this paper, we adopt a model for the hopping motion of parts as proposed by Boothroyd [5], [6] in the context of trap design, and extend our design algorithm to take this motion model into account. This extension allows us to eliminate trap designs that are undesirable for practical use.

We consider rigid three-dimensional polyhedral parts of a single type, all of which are assumed identical. We assume the position of the center of mass C to be known, as it dictates the stability of the part. While moving up the track, a part rests on the track in a stable pose; that is, the part rests on the track floor and against the track wall, where both contacts support C. The stable floor contact is the result of gravity, while a slight tilt of the feeder track assures a stable wall contact. Finally, the part is rejected when its center of mass is no longer supported by the track floor.

As in the earlier works on algorithmic trap design, we assume parts move along a flat track in a quasi-static manner without interfering with each other. Also, we adopt the assumption that rejected parts are always deflected off the vibratory track; i.e., during part rejection, we assume the part-track friction to be zero and the acceleration of the part due to gravity to infinite. Although these assumptions still leave a gap to be bridged for the industrial application of automated trap design, we believe that the presented work provides a model flexible enough to allow for relaxing the aforementioned idealization. The radius of the helical track is assumed large compared to the dimensions of the part, allowing us to approximate the section of the track as linear.

Our goal in this paper is to develop an algorithm for the automated design of V-traps. The algorithm is complete in the sense that it identifies all existing valid trap designs that feed the part. It receives as input the polyhedral part P along with its center of mass C. The algorithm either outputs the set Σ of all valid trap designs that feed P or reports that no such trap exist. In an additional step, we also also take as input the maximum distance a part can travel in one "hop", and output all valid trap designs that feed P while taking the more realistic motion model into account.

This paper is organized as follows. In section 2, we discuss the part-trap interaction in more detail and provide a number of definitions. In section 3, we first provide an intuitive explanation of our algorithm and continue with a discussion at a more detailed level. Section 4 explains the hopping part motion and discusses the necessary augmentations to the algorithm presented in section 3. We conclude in section 5 with a brief summary, several remarks and future work.

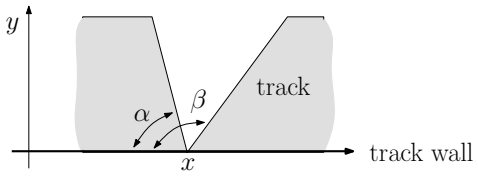

Fig. 2. Illustration of a trap placement $T(\alpha, \beta, x)$.

II. Modeling

We will first address the *track poses*, which are the possible poses of P whilst it moves up the track. The second subsection provides several general observations and definitions, and the third and final subsection discusses the interaction between part and trap.

For lack of space, we omit proofs of complexity and time bounds in the current and following section. These proofs can be found in section 2 and 3 of [12].

A. Track Poses

A polyhedral part in three-dimensional space has three rotational degrees of freedom. While moving up the track, P settles in a stable pose on the track, thus discretizing its set of possible orientations. A stable track placement consists of two stable contacts, namely the contact with the (track) floor and with the (track) wall. We assume the floor and wall are sufficiently wide and high, respectively, such that the both always contain the perpendicular projection of P.

A stable floor contact is a placement in which P rests on the floor with a face of its convex hull that supports C—the *convex hull* [9] of P is the smallest convex set of points containing P. A face is said to support C if, when resting on the floor, the face contains C projected along the direction of gravity onto the floor. Observe that such a contact discretizes two of the three degrees of rotational freedom.

The second aspect of a stable pose, the stable wall contact, is ensured by the earlier mentioned track tilt. This contact discretizes the remaining degree of freedom, the roll orientation, which is the rotation about the axis through C perpendicular to the floor contact of P. Given P in a stable floor contact, the discretization of the roll orientation depends on the shape of the perpendicular projection of P and C onto the track floor: a stable wall contact is a roll orientation in which the projection of P rests against the wall with an edge of its convex hull that supports the projected C.

In the remainder of this paper, we refer to a stable part pose simply as a pose. The number of possible part poses is $O(n^2)$, where n denotes the number of vertices of P.

B. Definition & Notation

Let us first look in more detail at T. In theory, the interval of α- and β-values, denoted by A and B, respectively, is $[0, \pi]$, where $\alpha < \beta$ (Fig. 2). In a practical setting, values in the lower and upper portions of these intervals produce traps

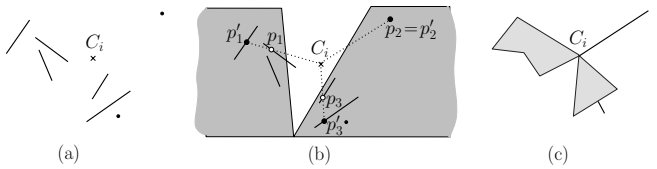

Fig. 3. (a) set of floor features of P_i touching the floor. (b) illustrates that it suffices to consider p'_1, p'_2 and p'_3 with respect to the safety of the trap placement. (c) star-hull of P_i.

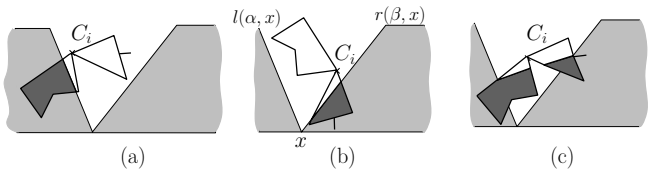

Fig. 4. P_i is (a) forward-unsafe, (b) backward-unsafe and (c) topple-unsafe.

that span too much track length to be practically feasible. We assume that the allowed α-interval, $A = [\alpha_s, \alpha_e]$, and β-interval, $B = [\beta_s, \beta_e]$, are given.

Let us consider P as it moves over T. The stability of a given pose positioned above T is determined by its floor features, which are the features (vertices and edges) of P resting on the floor when in the given pose. A pose is stable as long as the convex hull of the floor features contains the perpendicular projection of C onto the track floor. As the pose moves over T, each floor feature at some point temporarily loses contact with the floor. As a result, the pose can become unstable and gets rejected by T.

In summary, for a given pose, the stability above T and thus rejection by T is determined by its floor features; hence, the design algorithm only needs to consider the floor features of a pose. This observation reduces our problem to a planar one with respect to part-trap interaction (Fig. 3a). We denote the floor features of a pose by P_i, where $i \in \{1, \ldots, k\}$ and, as shown before, $k = O(n^2)$. Furthermore, we denote by n_i the number of vertices of P_i. We note that P_i can consist of faces, edges or vertices of P, or any combination of the aforementioned. For a given P_i, let C_i denote C projected along the direction of gravity onto the track floor.

Lemma 1. $\sum_{i=1}^{k} n_i = O(n^2)$.

We define the world coordinate frame such that the wall coincides with the horizontal x-axis, and the positive y-axis crosses the floor (Fig. 2). We note that $(x, 0)$ describes a point on the track wall. For ease of explanation, we consider the part as stationary and slide the trap underneath the part in the negative x-direction. We extend our trap notation and denote a *trap placement* by $T(\alpha, \beta, x)$, where $x \in X \subset \mathbb{R}$ specifies the position of the intersection point of both trap edges on the x-axis (Fig. 2). Furthermore, we denote a *left* and *right trap edge* by respectively $l(\alpha)$ and $r(\beta)$ and, similarly, use $l(\alpha, x)$ and $r(\beta, x)$ to denote the left and right trap edge for a particular trap placement.

C. Part-Trap Interaction

For a given $T(\alpha, \beta, x)$, we define the *supported subset* $S_i(\alpha, \beta, x)$ of P_i as the subset of P_i that is in contact with the track. Similarly, we define $S_i(\alpha, x)$ and $S_i(\beta, x)$ as the subsets of P_i that are in contact with the track adjacent to $l(\alpha, x)$ and $r(\beta, x)$, respectively. If the convex hull of $S_i(\alpha, \beta, x)$ contains C_i, then P_i is stable for $T(\alpha, \beta, x)$. We refer to such a placement as a *safe placement*; a placement is *unsafe* otherwise. For a given P_i, α and x, placement $T(\alpha, \beta_c, x)$ is a *critical placement* when $T(\alpha, \beta, x)$ is safe if and only if $\beta \leq \beta_c$. Alternatively, we say that β_c is the *critical* β for the given α and x.

Definition 1. *For a given P_i, α and x, the* critical β_c *is the β-value for which $T(\alpha, \beta, x)$ is a safe placement for $\beta \leq \beta_c$ and an unsafe placement for $\beta > \beta_c$.*

For a given P_i, a *safe trap* (design) $T(\alpha, \beta)$ is a trap for which each placement is safe; i.e., $T(\alpha, \beta)$ passes P_i without rejecting P_i. For a given P_i and α, we say that $T(\alpha, \beta_c)$ is a *critical trap* if it is a safe trap that features at least one critical placement; hence, $T(\alpha, \beta)$ is a safe trap for P_i for $\beta \leq \beta_c$, while unsafe for $\beta > \beta_c$. Again, we alternatively say that β_c is the critical β for the given α.

Definition 2. *For a given P_i and α, the* critical β_c *is the β-value for which $T(\alpha, \beta)$ is a safe trap for $\beta \leq \beta_c$ and an unsafe trap for $\beta > \beta_c$.*

The *star-hull* of P_i captures the part of P_i that is relevant to the safety of P_i over an arbitrary trap. The star-hull is defined by $\bigcup_{p \in P_i} \overline{C_i p}$ or, more informally defined, it is the set of points of P_i furthest away from C_i in every direction. Fig. 3a and fig. 3c show an example P_i and its star-hull, respectively. Intuitively, any triple of points in $S_i(\alpha, \beta, x)$ that defines a triangle containing, i.e., supporting C_i, can be replaced by a triple of points in the star-hull of P_i that similarly supports C_i; hence, it suffices to consider points in the star-hull of P_i when reviewing trap safety (see [12] for details). We assume for the remainder of this paper that each P_i is represented by its star-hull. We note that the star-hull has the same asymptotic complexity as P_i.

Definition 3. *The star-hull of P_i is defined as $\bigcup_{p \in P_i} \overline{C_i p}$.*

We can distinguish three types of unsafety that can cause P_i to fall into T: *forward-unsafety*, P_i falls forward (Fig. 4a) when C_i is positioned over T, but P_i is not yet supported by $r(\beta, x)$; *backward-unsafety*, P_i falls backward (Fig. 4b) when C_i is positioned over T, but P_i not is supported by $l(\alpha, x)$; and *topple-unsafety*, P_i topples (Fig. 4c) when C_i is positioned over T and P_i is in contact with both $l(\alpha, x)$ and $r(\beta, x)$, but C_i is unsupported.

III. Trap Design

In the first subsection, we provide an intuitive sketch of our algorithm. The second and third subsection model falling and toppling. The last subsection combines these two models to identify the set of all valid solutions Σ.

A. General Approach

A specific V-trap is defined by α and β. These parameters span a two-dimensional space, which we refer to as the *trap*

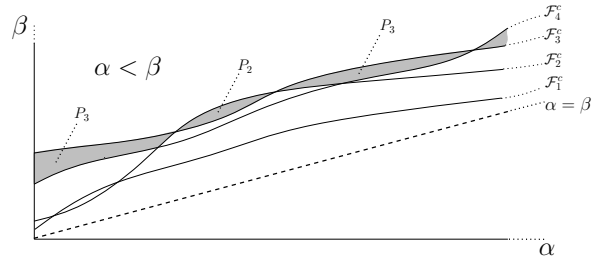

Fig. 5. This figure shows four arbitrary critical trap functions in a subset of the trap space (these example functions do not match any real part). The light gray area depicts Σ, where the fed poses are specified for three subsets of Σ.

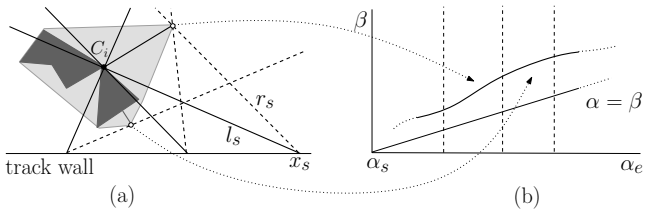

Fig. 6. (a) l (solid lines) for three x values and r (dotted lines) for the corresponding β_f. Note that l_s and r_s are shorthands for $l(\alpha_s, x_s)$ and $r(\beta_f, x_s)$, respectively. (b) illustrates the shape of a resulting graph—the graph does not match the part. The arrows between fig. (a) and (b) illustrate the relation between two convex hull points and their corresponding fall curve segments.

space; each point in this space describes a unique V-trap. The idea is to subdivide the trap space for each P_i into a safe and unsafe region, where a (un)safe region is a set of points corresponding to (un)safe traps for P_i. We then combine the subdivisions of all P_i and extract Σ. This subdivision for a given P_i is defined by the *critical trap function* $\mathcal{F}_i^c(\alpha)$, which specifies β_c for each α in $[\alpha_s, \alpha_e]$. The curve specified by this function divides the trap space in two subsets: points below or on $\mathcal{F}_i^c$ correspond to safe traps for P_i, while points above $\mathcal{F}_i^c$ correspond to traps that reject P_i.

The subdivisions are combined as follows. We add $\mathcal{F}_i^c$ for all $i \in 1, \ldots, k$ to the trap space, resulting in an arrangement of curves [9], [13]. The trap designs that reject all but one stable pose correspond to the points in the trap space that are above all but one $\mathcal{F}_i^c$. Let us refer to such a point as a valid solution σ and to the set of all valid solutions as Σ (Fig. 5).

Let us summarize, for this concept plays a central role in our algorithm. Each $\mathcal{F}_i^c$ corresponds to one specific P_i and specifies the impact of all possible trap designs on P_i. There exist $O(n^2)$ poses, thus the trap space contains $O(n^2)$ critical trap curves. By combining all $\mathcal{F}_i^c$, we are able to extract Σ.

We zoom in onto the critical trap functions, which are the building blocks of the algorithm. The computation of $\mathcal{F}_i^c$ consists of several algorithmic steps. Recall that we distinguish between forward-, backward- and topple-unsafety. We capture forward and backward-unsafety with the *fall function* $\mathcal{F}_i^f(\alpha)$, and we capture the topple-unsafety with the *topple function* $\mathcal{F}_i^t(\alpha)$. The function $\mathcal{F}_i^c$ is a composition of $\mathcal{F}_i^t$ and $\mathcal{F}_i^f$. These, as well as other functions defined in the following, are piecewise algebraic of bounded degree.

B. Falling

The fall function $\mathcal{F}_i^f$ specifies β_f for each α in $[\alpha_s, \alpha_e]$ such that P_i falls (does not fall) forward or backward into $T(\alpha, \beta)$ for $\beta > \beta_f$ ($\beta \leq \beta_f$); i.e., intuitively, all points below or on $\mathcal{F}_i^f$ correspond to trap designs that are forward- and backward-safe for P_i. The $\mathcal{F}_i^f$ function is composed of $\mathcal{F}_i^{ff}(\alpha)$, which captures forward falling, and $\mathcal{F}_i^{fb}(\alpha)$, which captures backward falling. In the following, we discuss the approach to construct $\mathcal{F}_i^{ff}$—we omit the similar construction of $\mathcal{F}_i^{fb}$—and conclude with the formation of $\mathcal{F}_i^f$.

Let us first consider α_s and identify the corresponding β_f. We choose $x = x_s$ such the left trap edge $l(\alpha_s, x_s)$ contains C_i: here, C_i is about to start moving over T, so the chance on forward falling is maximal. We then choose β_f such that the right trap edge $r(\beta_f, x_s)$ is tangent to the right side of P_i (Fig. 6a). Clearly, choosing a larger β results in P_i falling forward into $T(\alpha, \beta)$, while a smaller β would not be a boundary value between the forward-safe and unsafe β's.

The approach to construct $\mathcal{F}_i^{ff}$ is as follows. Starting at α_s, we apply a sweep approach [9] by increasing α until $\alpha = \alpha_e$, while keeping C_i contained in l and maintaining r as the tangent to P_i. As a result, r "slides" along the convex hull of P_i. Using basic geometry, we map each convex hull vertex encountered by r to a curve of constant degree; each curve defines $\mathcal{F}_i^{ff}$ for a sub-interval of $[\alpha_s, \alpha_e]$.

Lastly, to construct $\mathcal{F}_i^f$, we combine $\mathcal{F}_i^{ff}$ and $\mathcal{F}_i^{fb}$ by taking the lower envelope of $\mathcal{F}_i^{ff}$ and $\mathcal{F}_i^{fb}$. The lower envelope is the geometric construction comprised of all (partial) curves that are minimal with respect to β [9], [13]; i.e., all points below or on both $\mathcal{F}_i^{ff}$ and $\mathcal{F}_i^{fb}$ correspond to trap designs that are forward- and backward-safe.

Lemma 2. *The function $\mathcal{F}_i^f$ consists of $O(n_i)$ curves of constant degree and requires $O(n_i)$ computation time.*

C. Toppling

The topple function $\mathcal{F}_i^t$ specifies β_t for each α in $[\alpha_s, \alpha_e]$ such that P_i does not topple into $T(\alpha, \beta)$ if and only if $\beta \leq \beta_t$; intuitively, P_i does (not) topple into traps corresponding to points (below or on) above $\mathcal{F}_i^t$. We recall that topple-unsafety can per definition only occur when C_i is over $T(\alpha, \beta, x)$ and both $l(\alpha, x)$ and $r(\beta, x)$ intersect P_i. In the remainder of this subsection, we will implicitly assume that $T(\alpha, \beta, x)$ satisfies these criteria.

Let us start with an alternative view on the definition of a safe part placement. We define a fixed local coordinate system with C_i as its origin. Let $\rho(p) \in \Gamma = [0, 2\pi]$ be the counter-clockwise angle of the half-line extending from C_i through point $p \in P_i$ with the positive x-axis of this local coordinate system (Fig. 7a). For a given $T(\alpha, \beta, x)$, we say that $\gamma \in \Gamma$ is *(un)supported* if there exists (no) $p \in S_i(\alpha, \beta, x)$ such that $\rho(p)$ equals γ. A placement $T(\alpha, \beta, x)$ is safe if and only if there exists no interval $[\gamma_a, \gamma_b]$ of length $|\gamma_a, \gamma_b| > \pi$ that is unsupported, where $|\gamma_a, \gamma_b|$ specifies the length of the γ-interval measured in counter-clockwise direction from γ_a. Furthermore, an interval is considered unsupported if it contains no supported γ (Fig. 7a).

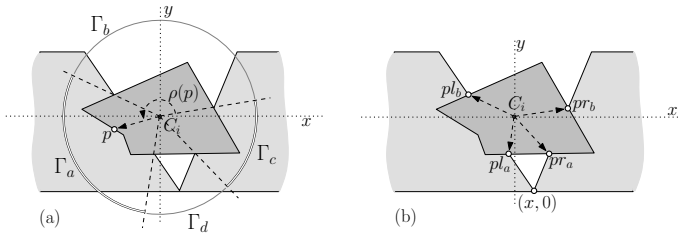

Fig. 7. (a) Part plus local coordinate system with C_i as origin; $\rho(p)$ specifies the angle in Γ for point p; intervals Γ_a, Γ_c and Γ_b, Γ_d are supported and unsupported subsets of Γ, respectively. (b) the intersection points pl_a and pl_b of $l(\alpha, x)$ with P_i, and pr_a and pr_b of $r(\beta, x)$ with P_i.

With the above definition of a safe placement in mind, let us revisit the topple-unsafety definition. For an arbitrary $T(\alpha, \beta, x)$, let pl_a and pl_b be the intersection points of $l(\alpha, x)$ with P_i with the smallest and largest distance to $(x, 0)$, respectively; furthermore, let pr_a and pr_b be defined similarly for $r(\beta, x)$ (Fig. 7b). We refer to pl_a and pr_a as the *first intersection point* of $l(\alpha, x)$ and $r(\beta, x)$ with P_i and, similarly, to pl_b and pr_b as the *last intersection points*.

Using these points, we can define four closed γ-intervals of which the extremes are certain to be supported, namely $[\rho(pr_a), \rho(pr_b)]$, $[\rho(pr_b), \rho(pl_b)]$, $[\rho(pl_b), \rho(pl_a)]$ and $[\rho(pl_a), \rho(pr_a)]$. We observe that $|\rho(pr_a), \rho(pr_b)| < \pi$ and $|\rho(pl_b), \rho(pl_a)| < \pi$. Thus, to determine whether $T(\alpha, \beta, x)$ is safe, we only need to consider $[\rho(pr_b), \rho(pl_b)]$ and $[\rho(pl_a), \rho(pr_a)]$. Keeping in mind that P_i is represented by its star-hull (Def. 3), it can be seen that the interiors of these two intervals contain no supported γ (see [12] for details).

Lemma 3. *A trap placement $T(\alpha, \beta, x)$ is topple-unsafe if and only if $|\gamma(pl_a), \gamma(pr_a)| > \pi$ or $|\gamma(pr_b), \gamma(pl_b)| > \pi$. The points pl_a and pl_b are the first and last intersection point of $l(\alpha, x)$ with P_i; pr_a and pr_b are defined similarly for $r(\beta, x)$.*

Following the above definition, we can distinguish between two types of topple-unsafety, corresponding to the event of $|\gamma(pl_a), \gamma(pr_a)| > \pi$ and $|\gamma(pr_b), \gamma(pl_b)| > \pi$. Intuitively, in the latter case, P_i rotates away from the wall and flips directly into the bowl, while in the former case, P_i attempts to rotate toward the wall and thus slides down along the wall back into the bowl. We model both types of topple-unsafety separately by constructing the functions $\mathcal{F}_i^{ta}$ and $\mathcal{F}_i^{tb}$, after which both functions are combined to form $\mathcal{F}_i^{t}$. We focus our discussion on $\mathcal{F}_i^{tb}$ that captures topple-unsafety related to pl_b and pr_b, i.e, the last intersection points of $l(\alpha, x)$ and $r(\beta, x)$ with P_i. The discussion of $\mathcal{F}_i^{ta}$ is very similar and is limited to a few remarks.

In contrast to the approach used to construct $\mathcal{F}_i^{f}$, we need to consider a range of x-values to determine $\mathcal{F}_i^{tb}(\alpha)$ for a given α. We introduce the *placement space*, which is a three dimensional space spanned by α, β and x; observe that each point in this space specifies a unique trap placement. In this space, we will construct an "extended" version of $\mathcal{F}_i^{tb}$ that takes x into account; we will denote this function by $\mathcal{G}_i^{tb}(\alpha, x)$. For a given α and x, this function specifies β_t such that P_i does not topple into $T(\alpha, \beta, x)$ if and only if $\beta \leq \beta_t$. Intuitively, $\mathcal{G}_i^{tb}$ describes a surface in the placement space such that P_i does (not) topple into traps corresponding to points (below or on) above $\mathcal{G}_i^{tb}$.

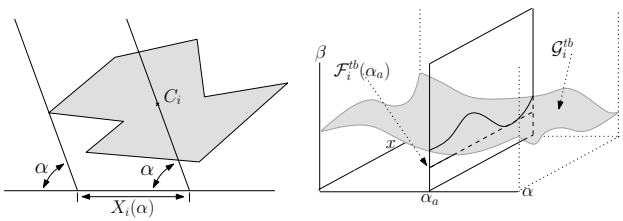

Fig. 8. (a) An example of $X_i(\alpha)$, (b) Illustration of mapping of $\mathcal{G}_i^{tb}$ to $\mathcal{F}_i^{tb}$.

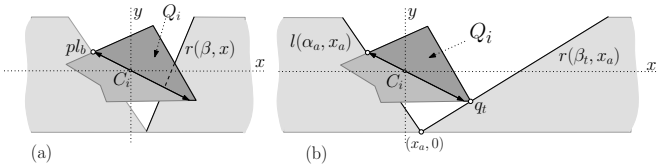

Fig. 9. (a) Example of Q_i that is intersected by $r(\beta, x)$. (b) Example of identification of β_t for (α_a, x_a)—$r(\beta_t, x_a)$ is tangent to Q_i in point q_t.

Let us first consider the domain of $\mathcal{G}_i^{tb}$. Recall that toppling can only occur when C_i is over T and P_i is in contact with both $l(\alpha, x)$ and $r(\beta, x)$; applied to $\mathcal{G}_i^{tb}$, this means that the domain of $\mathcal{G}_i^{tb}$ consists of all (α, x) for which $l(\alpha, x)$ intersects P_i and C_i is on the trap-side of $l(\alpha, x)$. We use $X_i(\alpha)$ to denote the set of x-values of this domain for a given α (Fig. 8). Let us consider $\mathcal{G}_i^{tb}$ and $\mathcal{F}_i^{tb}$ again. Clearly, P_i does not topple into $T(\alpha, \beta)$ when it does not topple into $T(\alpha, \beta, x)$ for $x \in X_i(\alpha)$. Hence, for a given α, $\mathcal{F}_i^{tb}$ is defined as the minimum of $\mathcal{G}_i^{tb}$ for $x \in X_i(\alpha)$ (see Fig. 8).

Definition 4. $\mathcal{F}_i^{tb}(\alpha) = \min\{\mathcal{G}_i^{tb}(\alpha, x) | x \in X_i(\alpha)\}$.

In summary, $\mathcal{G}_i^{tb}$ is the basis for the construction of $\mathcal{F}_i^{tb}$. We first discuss the identification of β_t for a given (α, x)-value, after which we continue by generalizing this concept and developing an algorithm to construct $\mathcal{G}_i^{tb}$. We conclude with the generation of $\mathcal{F}_i^{tb}$ using $\mathcal{G}_i^{tb}$.

We consider an arbitrary (α, x), denoted by (α_a, x_a), and determine the corresponding β_t. Let pl_b and pr_b be defined as before and let Q_i be defined as $\{p \in P_i | \rho(p) \in [\rho(pl_b) - \pi, \rho(pl_b)]\}$ (Fig. 9a). Based on lemma 3, we can conclude $T(\alpha_a, \beta, x_a)$ is topple-safe if and only if β is such that $r(\beta, x_a)$ intersects Q_i. Intuitively, Q_i contains all $p \in P_i$ for which $|\rho(p), \rho(pl_b)| \leq \pi$, hence β must be such that $r(\beta, x_a)$ contains a point of Q_i to ensure P_i does not topple into $T(\alpha_a, \beta, x_a)$. Upon reflection, we can conclude that β_t is such that $r(\beta_t, x_a)$ is tangent to Q_i. Let $q_t \in Q_i$ be the point where $r(\beta_t, x_a)$ touches Q_i (Fig. 9(b)). Clearly, for any $\beta > \beta_t$, $r(\beta, x_a)$ does not intersect Q_i, while for any $\beta < \beta_t$, $r(\beta, x_a)$ also intersecting $\overline{C_i q_t}$. As a final observation, we note that the convex hull of Q_i suffices to determine the tangent $r(\beta_t, x_a)$.

Definition 5. $Q_i(\gamma) = \mathcal{CH}(\{p \in P_i | \rho(p) \in [\gamma - \pi, \gamma]\})$, *where $\mathcal{CH}$ denotes the convex hull. Furthermore, let $vq_1(\gamma), vq_2(\gamma), \ldots, vq_m(\gamma)$ denote the vertices of $Q_i(\gamma)$, where m specifies the number of vertices in $Q_i(\gamma)$. The*

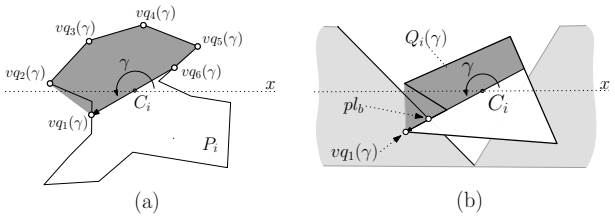

Fig. 10. (a). $Q_i(\gamma)$ for a given γ, where $vq_2(\gamma), \ldots, vq_5(\gamma)$ are fixed vertices of P_i and $vq_1(\gamma)$ and $vq_6(\gamma)$ depend on γ. (b). Example of an event for which $vq_1(\gamma) \neq pl_b$: point pl_b is the last intersection point of $l(\alpha, x)$, but is not equal to $vq_1(\gamma)$.

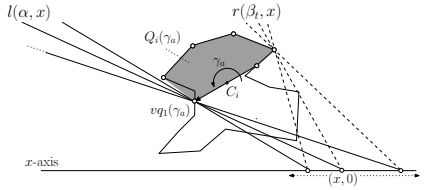

Fig. 11. By sliding $(x, 0)$ along the x-axis and maintaining $l(\alpha, x)$ through $vq_1(\gamma_a)$, construct $Q_i(\gamma_a)$ can be used to identify β_t for a set of (α, x) values.

vertices are numbered in counter-clockwise order and $\rho(vq_1(\gamma)) = \gamma$.

Let us further investigate $Q_i(\gamma)$ as this structure plays a central role in our approach. Firstly, we note that $vq_2, \ldots, vq_{m-1}$ are vertices of P_i and, except for their addition to and removal from $Q_i(\gamma)$, do not depend on γ. The position of vertices $vq_1(\gamma)$ and $vq_m(\gamma)$ does depend on γ (Fig. 10(a)). Secondly, when again considering (α_a, x_a), we can observe that pl_b is part of $Q_i(\gamma)$; more precise, their relation can be described as follows: $pl_b = vq_1(\rho(pl_b)) = vq_1(\gamma)$. For here on, we use $vq_1(\gamma)$ to refer to the last intersection point of $l(\alpha, x)$ with P_i. We note that there exists one exception to the latter note. That is, pl_b is not necessarily equal to $vq_1(\gamma)$ when the support line of an adjacent edge of $vq_1(\gamma)$ contains C_i (Fig. 10b). In the following discussion of the algorithm, we temporarily ignore such cases and discuss its treatment separately later on.

We generalize the use of $Q_i(\gamma)$. Let γ_a denote an arbitrary γ-value. Employing $Q_i(\gamma_a)$, we can determine β_t for all (α, x) for which $vq_1(\gamma_a)$ is the last intersection point of $l(\alpha, x)$ with P_i. Intuitively, this set of (α, x) values can be discovered by sliding a point $(x, 0)$ along the x-axis, while maintaining $l(\alpha, x)$ through $vq_1(\gamma_a)$ and letting $r(\beta_t, x)$ "glide" along the boundary of $Q_i(\gamma_a)$ (Fig. 11).

The above observation is the basis for the algorithm to construct $\mathcal{F}_i^{tb}$. The general idea of the algorithm is as follows. Let us first consider a given γ. We can identify the set of (α, x)-values for which $vq_1(\gamma)$ is the last intersection point of $l(\alpha, x)$. We observe, however, that it suffices for the algorithm to maintain the x-component of each (α, x); namely, for a given γ and x, there exists only one α such that $l(\alpha, x)$ contains $vq_1(\gamma)$. We refer to the resulting set of x values as the *active subset* of X (Fig. 12a and b). Subsequently, the algorithm determines β_t for each x in the active subset. In summary, for a given γ, the algorithm

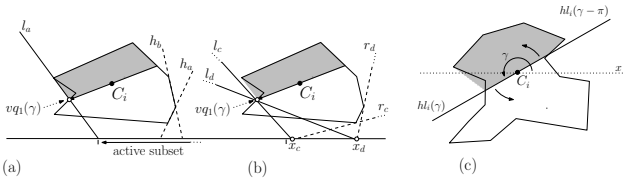

Fig. 12. (a). An example of the identification of an active subset of X, which in this case goes to infinity for $+x$. Furthermore, h_a and h_b give an impression of how β_t-values will be linked to active x-values. (b). (α_c, x_c) and (α_d, x_d) are two examples of active (α, x) values for which β_t is defined by r_c and r_d. (c). $hl_i(\gamma)$ and $hl_i(\gamma - \pi)$ sweeping over Pi.

models topple-unsafety by using $Q_i(\gamma)$ to identify the active subset of X and determine the corresponding β_t-values.

The above concept generalizes as follows. For an arbitrary initial γ_s, the algorithm constructs $Q_i(\gamma_s)$. Starting from γ_s, the algorithm then traverses Γ while maintaining $Q_i(\gamma)$. Simultaneously, it uses $Q_i(\gamma)$ to identify the active subset of X and the corresponding β_t-values for each visited γ. As a result, the algorithm constructs a planar subdivision [9] of the Γ, X-space. This subdivision specifies for each (γ, x) whether it is active and, if so, the corresponding β_t. Using the property that an active (γ, x) uniquely maps to a (α, x), the algorithm maps the Γ, X-space subdivision to the A, X-space, after which $\mathcal{G}_i^{tb}$ can be constructed using A, X-space subdivision and the stored β_t information. Finally, we can construct $\mathcal{F}_i^t(\alpha) = \min\{\mathcal{G}_i^{tb}(\alpha, x) | x \in X_i(\alpha)\}$ (Def. 4) by projecting $\mathcal{G}_i^{tb}$ onto the A, B-plane and computing the lower envelope of the resulting arrangement.

For lack of space, a detailed explanation of the algorithm is omitted. We refer the interested reader to section 3 of [12].

D. Solution

The composition of the trap function $\mathcal{F}_i^c$ by combining $\mathcal{F}_i^f$ and $\mathcal{F}_i^t$ is straightforward. Recall that all points on or below $\mathcal{F}_i^f$ and $\mathcal{F}_i^t$ correspond to trap designs that ensure P_i does not fall forward/backward nor toppling, respectively. Hence, intuitively, all points that are on or below both $\mathcal{F}_i^f$ and $\mathcal{F}_i^t$ correspond to safe traps for P_i. The function $\mathcal{F}_i^c$ is the lower envelope of $\mathcal{F}_i^f$ and $\mathcal{F}_i^t$.

Lemma 4. *$\mathcal{F}_i^c$ has a $O(n_i^2 2^{\alpha(n_i^2)})$ complexity and requires $O(n_i^2 2^{\alpha(n_i^2)})$ time to compute, where $\alpha(n)$ is the extremely slowly growing inverse of the Ackermann function.*

We generate $\mathcal{F}_i^c$ for each $i \in \{1, \ldots, k\}$ and add the result to the trap space, resulting in an arrangement of curves. We recall that a point that lies above all but one $\mathcal{F}_i^c$ in terms of β represents a valid solution; that is, it represents a V-trap that rejects all but one pose of P. The desired Σ is the part of trap space between the upper envelope and the critical curves one level down, where the upper envelope is the curve comprised of all (partial) critical curves that are maximal with respect to β [9]. In geometry, this part is referred to as the ≤ 1-*level* of an arrangement of curves [9] (Fig. 5).

Once Σ is generated, the algorithm reports the set of valid V-trap designs. Solution set Σ consists of a number of subsets, each consisting of valid solutions that feed a specific pose of P. No valid solution exists when the $\leq$ 1-level

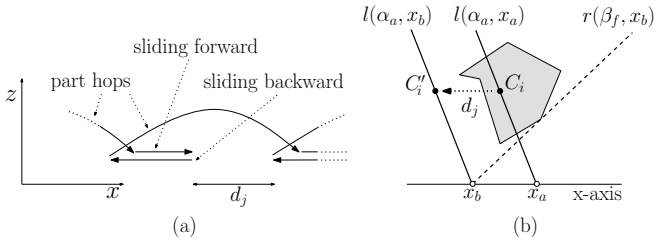

Fig. 13. (a) illustrates a possible part motion modus, where d_j is the effective jump distance. (b) shows placements $l(\alpha_a, x_a)$ and $l(\alpha_a, x_b)$ of the left trap edge, where $l(\alpha_a, x_b)$ is the placement at the moment P_i lands after jumping when C_i crosses $l(\alpha_a, x_a)$. Using C_i', which is C_i offset by d_j in negative x-direction, β^f as output by $\overline{\mathcal{F}}_i^{ff}(\alpha_a)$ can be computed.

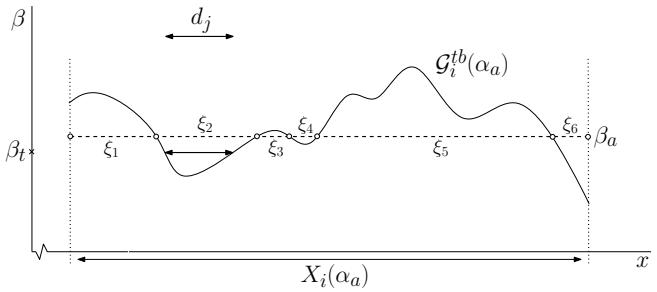

Fig. 14. shows $\mathcal{G}_i^{tb}$ for fixed $\alpha = \alpha_a$, defined for x-interval $X_i(\alpha_a)$. Set $\Xi(\alpha_a, \beta_a)$ consists of $\xi_1, \ldots, \xi_6$, and $\Xi^u(\alpha_a, \beta_a)$ consists of ξ_2, ξ_4 and ξ_6. Finally, the value of $\beta_t = \overline{\mathcal{F}}_i^{tb}(\alpha_a)$ is illustrated—observe for example that P_i is rejected by $T(\alpha_a, \beta_a)$ for ξ_2 is in $\Xi^u(\alpha_a, \beta_a)$ and $|\xi_2|_x > d_j$.

is empty; in other words, when the entire upper envelope consists of coinciding critical curves. The occurrence of the latter is unlikely. We conjecture that V-traps do not exist for certain classes of parts with prominent geometric symmetries; characterizing the class of parts for which V-traps are guaranteed to exist is a subject for future research.

Theorem 1. *The complexity of Σ is $O(a2^{\alpha(a)})$ and Σ can be constructed in $O(a2^{\alpha(a)} \log a)$ time, where $a = n^3 2^{\alpha(n^3)}$.*

IV. Modeling Part Motion

Abandoning the assumption that parts glide along the track, the following considers the hopping part motion induced by the asymmetric helical vibration of the vibratory bowl. Fig. 13(a) illustrates a motion modus proposed by Boothroyd [5], who applies this model among others in the context of trap design. The model can be explained as follows. In one vibration cycle, the upward motion of the bowl accelerates P, after which P is launched into flight as the bowl starts to move back. Next, P lands and slides forward a bit due to inertia, and then slides back a bit again as the bowl returns to its upward motion. Depending on the part and bowl, different part motion modes are possible as well. Still, all modes have in common that the part has some effective *jump distance*, denoted by d_j, which we assume to be known. As a result, P_i may unjustly survive $T(\alpha, \beta)$, hence T does not satisfy the feeding property.

This section discusses the computation of the set of all trap designs that feed P while taking the above introduced motion model into account. We remark that this solution set, denoted by $\overline{\Sigma}$, effectively consists of two subsets: $\overline{\Sigma} \cap \Sigma$ and $\overline{\Sigma} - \Sigma$. Let $\overline{\sigma}_a$ be an arbitrary trap design in $\overline{\Sigma}$, and let P_a be the pose fed by $\overline{\sigma}_a$. Trap $\overline{\sigma}_a$ feeds all occurrences of P_a when $\overline{\sigma}_a$ is in $\overline{\Sigma} \cap \Sigma$: trap $\overline{\sigma}_a$ is in Σ thus all trap placements of $\overline{\sigma}_a$ are safe for P_a. Whereas $\overline{\sigma}_a$ feeds only a percentage of the occurrences of P_a when $\overline{\sigma}_a$ is in $\overline{\Sigma} - \Sigma$: certain placements of $\overline{\sigma}_a$ are unsafe for P_a, therefore the specific jump pattern of a given occurrence of P_a determines its rejection.

The algorithm to compute $\overline{\Sigma}$ is similar to the (original) algorithm that computes Σ. In overview, the algorithm to compute $\overline{\Sigma}$ constructs functions that capture forward- and backward-unsafety and both types of topple-unsafety, while taking the jumping motion into account; let these functions be denoted by $\overline{\mathcal{F}}_i^{ff}$, $\overline{\mathcal{F}}_i^{fb}$, $\overline{\mathcal{F}}_i^{ta}$ and $\overline{\mathcal{F}}_i^{tb}$. The subsequent construction of critical trap functions and solution set $\overline{\Sigma}$ is the same as for the original algorithm.

The complexity of $\overline{\Sigma}$ and its computation time can be shown to be polynomial; further research is needed to determine the precise specification of these (polynomial) bounds.

A. Falling

This subsection addresses the computation of $\overline{\mathcal{F}}_i^{ff}$—the similar construction of $\overline{\mathcal{F}}_i^{fb}$ is omitted. We show will that with a minor adjustment, the original algorithm to compute $\mathcal{F}_i^{ff}$ (section III-B) can also be applied to compute $\overline{\mathcal{F}}_i^{ff}$.

We consider an arbitrary α-value, denoted by α_a, and determine $\beta_f = \overline{\mathcal{F}}_i^{ff}(\alpha_a)$. As in the original approach, we select $x = x_a$ such that $l(\alpha_a, x_a)$ contains C_i: this trap placement—pose P_i is about to fall forward—offers the best chance for P_i to escape rejection. Assuming the extreme case that P_i indeed jumps at trap placement $x = x_a$, then T moves by d_j in the negative x-direction before P_i again lands. Let $x_b = x_a - d_j$ denote the new trap placement. Finally, following the original approach, angle β_f is such that $r(\beta_f, x_b)$ is tangent to P_i (Fig. 13b).

In short, the original approach to identify β_f for α_a remains unchanged, except for the intermediate offset of the trap placement. Instead of adding this intermediate step to the original algorithm, we can offset C_i by d_j in negative x-direction (Fig. 13b). Choosing the left trap edge for α_a such that it contains C_i then directly produces x_b and subsequently β_f. In conclusion, after offsetting C_i, the algorithm presented in section III-B can be applied to compute $\overline{\mathcal{F}}_i^{ff}$.

B. Toppling

As $\mathcal{F}_i^{tb}$ for a given P_i, function $\overline{\mathcal{F}}_i^{tb}$ is derived from the $\mathcal{G}_i^{tb}$ (section III-C). In the following, we first explain the identification of $\beta_t = \overline{\mathcal{F}}_i^{tb}(\alpha)$ for a given α and then present an algorithm that employs this idea to compute $\overline{\mathcal{F}}_i^{tb}$.

Let us consider the cross-section of $\mathcal{G}_i^{tb}$ for a fixed $\alpha = \alpha_a$, denoted by $\mathcal{G}_i^{tb}(\alpha_a)$. In the X, B-plane containing $\mathcal{G}_i^{tb}(\alpha_a)$, we examine the horizontal line segment at fixed $\beta = \beta_a$ that spans x-interval $X_i(\alpha_a)$—observe that this line segment describes all trap placements in which C_i is over $T(\alpha_a, \beta_a)$. The set of intersections with $\mathcal{G}_i^{tb}(\alpha_a)$ subdivides the line segment into a set of sub-segments, which we denote by $\Xi(\alpha_a, \beta_a)$ (Fig. 14).

We observe that the interior of a given segment in $\Xi(\alpha_a, \beta_a)$ is either below or above $\mathcal{G}_i^{tb}(\alpha_a)$, thus describes a connected set of trap placements that are either safe or unsafe for P_i, respectively. Let the set of segments above $\mathcal{G}_i^{tb}(\alpha_a)$ be denoted by $\Xi^u(\alpha_a, \beta_a)$. Furthermore, we observe that by increasing or decreasing β, the length of the segments in $\Xi^u(\alpha_a, \beta)$ monotonically increases or decreases, respectively.

Angle $\beta_t = \overline{\mathcal{F}}_i^{tb}(\alpha_a)$ is defined as the maximum β-value for which the length of each segment in $\Xi^u(\alpha_a, \beta)$ is at most d_j. Intuitively, for any $\beta \leq \beta_t$, the aforementioned monotonicity ensures that the intervals of unsafe trap placements are short enough for P_i to be able to jump across, while for any $\beta > \beta_t$, there exists at least one interval of unsafe trap placements that causes P_i to topple into T.

Definition 6. $\overline{\mathcal{F}}_i^{tb}(\alpha) = \max\{\beta \mid \forall \xi \in \Xi^u(\alpha, \beta) : |\xi|_x \leq d_j\}$, *where* $|\xi|_x$ *is the length of the* x*-interval spanned by* ξ.

To compute $\overline{\mathcal{F}}_i^{tb}$, we need to express the algorithm as an operation on $\mathcal{G}_i^{tb}$. Again considering $\mathcal{G}_i^{tb}(\alpha_a)$ for fixed $\alpha = \alpha_a$, we sketch the approach to compute $\beta_t = \overline{\mathcal{F}}_i^{tb}(\alpha_a)$. Let the *unsafe segment* of a point p in $\mathcal{G}_i^{tb}(\alpha_a)$ be defined as the longest horizontal line segment that contains p and consists solely of points in and above $\mathcal{G}_i^{tb}(\alpha_a)$. The unsafe segment of p is *critical* either if its length equals d_j, or if its length is larger than d_j and the neighboring points of p in $\mathcal{G}_i^{tb}(\alpha_a)$ feature unsafe segments with lengths smaller than d_j—the latter occur only if p is a local maximum of $\mathcal{G}_i^{tb}(\alpha_a)$. To compute β_t, the algorithm identifies the *critical point set*, which is the set of points in $\mathcal{G}_i^{tb}(\alpha_a)$ featuring critical unsafe segments; β_f is the minimum β of the critical point set.

Generalizing over $[\alpha_s, \alpha_e]$, function $\overline{\mathcal{F}}_i^{tb}$ can be computed by employing a plane-sweep approach [9], [13], which can be sketched as follows. The sweep algorithm starts at α_s, where it computes the critical point set of $\mathcal{G}_i^{tb}(\alpha_s)$, orders the points in the critical point set on their β-value, and reports the minimum β-value. Then, while increasing α, the algorithm maintains the critical point set of $\mathcal{G}_i^{tb}(\alpha)$ and its β-ordering. Events—appearances and disappearances of points in the critical point set and changes in the β-ordering of the critical point set—occur at a discrete set of α-values. Function $\overline{\mathcal{F}}_i^{tb}$ is described by the set of reported minimum β-values over $[\alpha_s, \alpha_e]$.

V. Conclusion

We have presented an algorithm for the automated design of V-shaped traps for vibratory bowl feeder—these devices receive a stream of identical polyhedral parts in arbitrary stable pose as input and output parts in one single pose. Inspired by similar devices used in existing vibratory feeder systems, the proposed work on V-traps serves as a representative study to feeding 3D parts through rejection of part poses. Under the assumption that the motion of the part is quasi-static and rejected parts are always deflected off the track, our complete algorithm reports either all possible single V-trap solutions or that no solution exists. In addition, we have removed a common idealized assumption on the motion of the parts and replaced it by a more realistic model for part motion; this model has been incorporated in our design algorithm.

Our aim in this paper has been to take the design of complete algorithms for trap design to the realm of three-dimensional parts. A challenging open problem is to fully characterize the class of parts that are feedable with a single V-trap. Such a classification can either motivate or rule out the necessity of a study of more complex trap shapes and their design algorithms.

Another open problem is the interaction between the part and trap. Both previous and the herein presented research on algorithmic trap design assume that parts move in a quasi-static manner and rejected parts are always reflected off the track. No research has been done to date on the relaxations of these idealizations.

References

[1] P. Agarwal, A. Collins, and J. Harer. Minimum trap design. *IEEE ICRA*, pages 2243–2248, 2001.

[2] D. Berkowitz and J. Canny. Designing part feeders using dynamic simulation. *IEEE ICRA*, pages 1127–1132, 1996.

[3] R.-P. Berretty, K. Goldberg, M. Overmars, and A. van der Stappen. Trap design for vibratory bowl feeders. *International Journal of Robotics Research*, 20:891–908, 2001.

[4] R.-P. Berretty, M. Overmars, and A. van der Stappen. Orienting polyhedral parts by pushing. *Comput. Geom.*, 21(1-2):21–38, 2002.

[5] G. Boothroyd. *Assembly Automation and Product Design*. Taylor & Francis Ltd, 2005.

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

[7] M. Caine. The design of shape interactions using motion constraints. *IEEE ICRA*, pages 366–371, 1994.

[8] A. Christiansen, A. Edwards, and C. Coello. Automated design of parts feeders using a genetic algorithm. *IEEE ICRA*, pages 846–851, 1996.

[9] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry – Algorithms and Applications*. 1997.

[10] O. Goemans, K. Goldberg, and A. van der Stappen. Blades: A geometric primitive for feeding 3d parts on vibratory tracks. *IEEE ICRA*, pages 1730– 1736, 2006.

[11] O. Goemans, A. Levandowski, K. Goldberg, and A. van der Stappen. On the design of guillotine traps for vibratory bowl feeders. *IEEE CASE*, pages 79–86, 2005.

[12] O. Goemans and A. van der Stappen. On the design of traps for feeding 3d parts on vibratory tracks. *Technical report UU-CS-2007-028*, 2007.

[13] J. Goodman and J. Rourke, editors. *Handbook of Discrete and Computational Geometry (Second Edition)*. 2004.

[14] M. Jakiela and J. Krishnasamy. Computer simulation of vibratory part feeding and assembly. 2^{nd} *Int. Conf. on Discrete Element Methods*, pages 403–411, 1993.

[15] L. Lim, B. Ngoi, S. Lee, S. Lye, and P. Tan. A computer-aided framework for the selection and sequencing of orientating devices for the vibratory bowl feeder. *Int. J. of Production Research*, 32(11):2513–2524, 1994.

[16] K. Lynch. Inexpensive conveyor-based parts feeding. *Assembly Automation Journal*, 19(3):209–215, 1999.

[17] M. Mason. *Mechanics of Robotic Manipulation*. MIT Press, 2001.

[18] G. Maul and M. Thomas. A systems model and simulation of the vibratory bowl feeder. 2^{nd} *International Conference on Discrete Element Methods*, 16(5):309–314, 1997.

[19] J. Selig and J. Dai. Dynamics of vibratory bowl feeders. *IEEE ICRA*, pages 3299–3304, 2005.

[20] R. Silversides, J. Dai, and L. Seneviratne. Force analysis of a vibratory bowl feeder for automatic assembly. *ASME: Journal of Mechanical Design*, 127(4):637–645, 2005.

[21] M. Tay, P. Chua, S. Sim, and Y. Gao. Development of a flexible and programmable parts feeding system. *International Journal of Production Economics*, 98(2):227–237, 2005.

[22] R. Zhang and K. Gupta. Automatic orienting of polyhedra through step devices. *IEEE ICRA*, pages 550–556, 1998.