Robotics: Science and Systems 2005
Cambridge, MA, USA, June 8-11, 2005

A polynomial-time algorithm to design push plans for sensorless parts sorting

Mark de Berg*, Xavier Goaoc* and A. Frank van der Stappen†
* Department of Mathematics and Computer Science
TU Eindhoven, Eindhoven, The Netherlands.
Email: m.t.d.berg@tue.nl, goaoc@loria.fr
† Department of Information and Computing Sciences
Utrecht University, PO Box 80089, 3508 TB Utrecht, The Netherlands.
Email: frankst@cs.uu.nl

***Abstract*— We consider the efficient computation of sequences of push actions that simultaneously orient two different polygons. Our motivation for studying this problem comes from the observation that appropriately oriented parts admit simple sensorless sorting. We study the sorting of two polygonal parts by first putting them in properly selected orientations. We give an $O(n^2 \log n)$-time algorithm to enumerate all pairs of orientations for the two parts that can be realized by a sequence of push actions and admit sensorless sorting. We then propose an $O(n^4 \log^2 n)$-time algorithm for finding the shortest sequence of push actions establishing a given realizable pair of orientations for the two parts. These results generalize to the sorting of k polygonal parts.**

I. Introduction

Designers of robotic manipulators for factory environments have long been inspired by human arms and hands. The wish for a level of flexibility comparable to that of a human arm and hand led to robotic manipulators that were often found to be too complex to have a chance to stand up in a real industrial environment [18]. In the late 80s and early 90s, Whitney [44] and others argued that effective factory robots need far less flexibility than human beings. As more flexibility incurs an increase of design and maintenance costs, risk of failure, and complexity of control, it is justified to be cautious about excess flexibility. Inspired by Whitney's recommendations that industrial robots should have simple actuators and sensors Canny and Goldberg [13], [14] proposed the Reduced Intricacy in Sensing and Control (RISC) paradigm for the design of manipulation systems. The paradigm favors easily-reconfigurable simple hardware elements performing simple actions over overly flexible general-purpose hardware, and prefers simple or no sensors. The authors argued that such systems are cheaper, more reliable and better suited for automated planning.

RISC implies a shift of the complexity of system design to computer science, as the fundamental question becomes algorithmic in nature: configure, or plan, a sequence of simple physical actions that accomplishes a higher level manipulation task on a given part or collection of parts. Over the past years researchers have explored the suitability of sequences of actions such as pushing [1], [2], [4], [5], [7], [10], [16], [23], [25], [27], [30], [31], [45] squeezing [11], [15], [19], [34], [32], [33], [35], toppling [24], [46], pulling [6], tapping [21], dropping [20], [22], [29] wobbling [17], rolling [26], and vibrating [8], [9], [37] by simple hardware elements to accomplish a common task like feeding (or orienting) parts. Considerable attention has also been given to the design of modular fixtures [12], [38], [39], [40], [43], [41], [42], [47] which hold parts using simple reuable elements whose placements are constrained to a grid of holes. Much less has been done on some of the other challenges listed in the paper by Canny and Goldberg [14]. In this paper we study their open problem concerning the existence of a polynomial-time algorithm for sorting parts with a frictionless parallel-jaw gripper capable of performing push and squeeze actions [18].

We consider the following completely sensorless sorting scenario for convex polygonal parts of two different types P and Q that can be in any orientation. We have a conveyor belt that forks into two smaller sub-belts, as in Figure 1, and we want to employ push actions by a single jaw of the parallel-jaw gripper to establish that parts of type P continue on one sub-belt at the split while those of type Q continue on the other sub-belt. Which sub-belt a part goes depends on the position of its center of mass with respect to the line through the split and parallel to the sides of (the wide part of) the belt. As a result, our sorting scenario is successful if we can find a sequence of push actions, or *push plan*, that—at the same time—puts the center of mass of parts of type P on one side of a horizontal line and of parts of type Q on the other side. (We assume that the parts are sufficiently far apart when they travel down the belt, so that the orientations are not disturbed by any collisions.) We will concentrate on push plans that additionally bring parts of the same type into the same orientation. Our goal is therefore to find the shortest push plan that simultaneously puts P and Q into given orientations ϕ and ψ respectively; for our application, the orientations ϕ and ψ should be such that the distances from the jaw to the centers of mass of P and Q are different. Once P and Q are in their

Part of this research has been funded by the Dutch BSIK/BRICKS project. Mark de Berg and Xavier Goaoc were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

respective orientations, a single push by the jaw suffices to put P and Q onto the belt, after which the split will take care of the sorting. Non-convex parts traveling with a cavity facing the split may get stuck at the joint corner of the sub-belts. We can easily identify the orientations in which a part can get stuck beforehand and eliminate those from consideration in the planning phase.

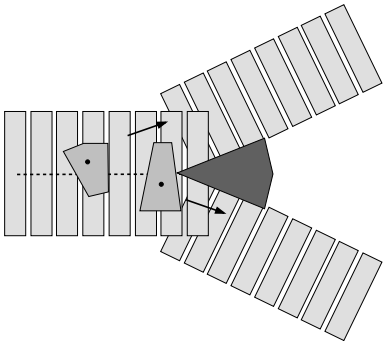

Fig. 1. A conveyor belt forking into two sub-belts and two parts that will be sorted.

Goldberg [19] showed that a single polygonal part with n vertices can be oriented, i.e., brought into any priorly specified orientation, by a push plan; he also showed that the shortest such plan can be computed in $O(n^2)$ time. Chen and Ierardi [15] proved that the length of the shortest plan is $O(n)$. Although a concatenation of separate plans for parts P and Q puts both parts into a known orientation, these results do not provide a way of finding the *shortest* sequence of push actions that simultaneously puts P and Q into *specified* orientations ϕ and ψ. In fact, one of the first results in this paper shows that it is not always possible to put P and Q in any desired combination of orientations. For polygons P and Q with at most n vertices, we therefore present an $O(n^2 \log n)$-time algorithm to determine all combinations of orientations for polygons P and Q that can be realized by applying an oblivious push plan to both. If none of these combinations separates the centers of mass then no push plan exists that sorts the parts. The main result of our paper is an algorithm that finds the shortest push plan that puts P and Q in a given realizable pair (ϕ, ψ) of orientations. The algorithm runs in $O(n^4 \log^2 n)$, so we have a polynomial-time algorithm for sorting using a pushing jaw and a forking conveyor belt. The availability of all realizable pairs of orientations offers the practical advantage of being able to choose the pair that maximizes the separation between the centers of mass, as such a pair is likely to be less sensitive to control errors and part imprecision.

The algorithms for determining all realizable combinations of orientations and for computing the shortest plan for a given realizable combination generalize to k parts. The running times of these generalizations are $O(n^k \log n)$ and $O(n^{2k} \log^{2k} n)$ respectively. All of our algorithms use a multi-dimensional generalization of a simplified version of the push function (see e.g. [19], [34]). The problems at hand are translated into geometric queries that are solved efficiently using geometric data structures [3].

Our work is closest in spirit to two papers by Rao and Goldberg [32], [36]. In [32] these authors considered the problem of recognizing a part from a given set of parts, by means of a sequence of width measurements by a squeezing instrumented parallel jaw gripper. Rao and Goldberg proposed an $O(n^4 2^n)$-time algorithm for computing the shortest sequence of measurements, and an $O(n^2 \log n)$-time algorithm for computing a sub-optimal sequence, where n is the total number of stable diameters, which is upper bounded by the total number of edges of all parts. This sorting scenario differs from ours in several aspects. Besides that we use push instead of squeeze actions, we have replaced the instrumentation of the gripper (facilitating the width measurements) by a forking conveyor belt. This replacement of sensing functionality by an additional piece of hardware opens the way to a polynomial-time algorithm for finding the shortest sequence of actions. Still, we could apply our algorithm to the instrumented parallel jaw gripper of Rao and Goldberg. If we choose the final orientations to be such that a squeeze action of the gripper leads to different width measurements for the two parts, then our algorithm computes in $O(n^4 \log^2 n)$ time the shortest sequence of push actions followed by a squeeze that orients and sorts the parts.

In [36], Rao and Goldberg studied the registration mark problem. Given a single polygon with n vertices and a set of k possible poses of that polygon, it asks to locate a mark on the polygon that maximizes the minimum separation between the placements of the mark in the k poses. The computed location will provide maximum insensitivity to sensor noise when distinguishing the poses with a computer vision system. The registration mark problem is in a way almost dual to our problem. Whereas the registration mark problem asks to determine the optimal distinguishing aspect, the mark, of a given set of poses, our problem is to determine the poses (and the actions that lead to these poses) that optimize the ability to distinguish on the basis of a given aspect, the center of mass. Of course another crucial difference is that we are dealing with two (or more) parts, whereas the work of Rao and Goldberg deals with a single part.

II. Preliminaries

In this section we briefly review pushing of a single polygon P by a jaw of a parallel-jaw gripper. We assume zero friction between the part and the jaw. Let c be the center-of-mass of P. As a jaw always touches the part at its convex hull we assume that P is convex. For the sake of simplicity of our analysis, we add the weak assumption that no line through a vertex v of P is perpendicular to the two edges incident to v. This prevents so-called meta-stable edges.

We assume that a fixed coordinate frame is attached to P. Directions are expressed relative to this frame. The *contact direction* of a tangent l of P is uniquely defined as the direction of the vector perpendicular to l and pointing into P (see Figure 2 for a tangent with contact direction π.). As in

Mason [27], we define the *radius function* $\rho : [0, 2\pi) \to \mathbb{R}_+$ of P with center of mass c; ρ maps a direction ϕ onto the distance from the c to the tangent of P with contact direction ϕ. The radius function is continuous. It determines the push function, which, in turn, determines the final orientation of a part that is being pushed.

Throughout this paper, parts are pushed by a single jaw that moves in a direction perpendicular to itself. Brost [11] was the first to model parallel-jaw gripper motions in this manner. The *push direction* of a single jaw is the direction of its motion. The push direction of a jaw pushing a part equals the contact direction of the jaw. In most cases, parts will start to rotate when pushed. If pushing in a certain direction does *not* cause the part to rotate, then we refer to the corresponding direction as an *equilibrium (push) direction* or *orientation*. Equilibrium orientations play a key role throughout this paper. If pushing does change the orientation, then this rotation changes the orientation of the pushing jaw with respect to the frame attached to the part. We assume a push action to be a reorientation of the jaw followed by an actual push on the object that continues until the part stops rotating and settles in a stable equilibrium pose.

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

The push function p for a polygonal part consist of *steps*, which are open intervals $I \subset [0, 2\pi)$ on which $p(\phi)$ is constant, and isolated fixed points. Each step of the push function intersects the diagonal line through the origin at the equilibrium orientation corresponding to the step. The steps of the push function are easily constructed [19] from the radius function ρ, using its local extrema; the orientations corresponding to local extrema are the equilibrium push orientations. If the part is pushed in a direction that is not a local extremum of the radius function then the part will rotate in the direction in which the radius decreases until it finally settles in an orientation corresponding to a local minimum of the radius function. As a result, all points in the open interval I bounded by two consecutive local maxima of the radius function ρ map onto the orientation $\phi \in I$ corresponding to the unique local minimum of ρ on I. (Note that ϕ itself maps onto ϕ because it is a local extrema.) The fixed points of the steps are the *stable* equilibrium orientations. Besides the steps and ramps there are isolated points satisfying $p(\phi) = \phi$ in the push function, corresponding to local maxima of the radius function. The isolated fixed points are the *unstable* equilibrium orientations.

Projecting the steps of the push function on the horizontal axis we get the *push diagram*: a partition of the range $[0, 2\pi)$ into open intervals by the unstable equilibrium orientations along with a collection of stable equilibriums, one per interval. A push action can be visualized in the push diagram by applying to the original orientation of the part a translation and a snap rounding. The length of the translation is the pushing angle and the rounding brings the translated point to the stable equilibrium of the interval that contains it. Figure 2 shows an example of a radius function and the corresponding push function.

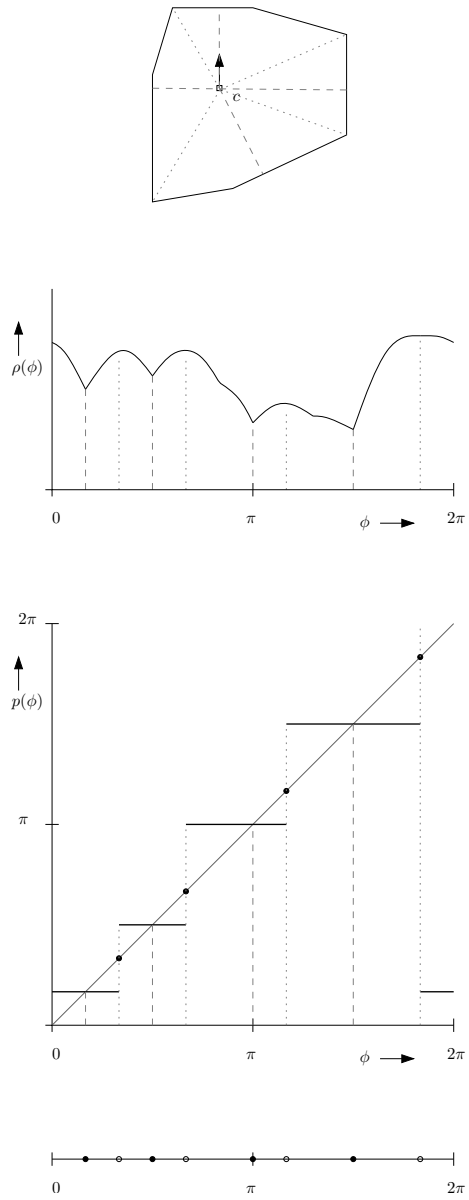

Fig. 2. A part, its radius function, the corresponding push function and the simplified push diagram. The vector emanating from the center-of-mass c shows the zero contact direction for supporting lines.

In the rest of this paper, P and Q denote two polygonal shapes, with p and q vertices respectively. To simplify the bounds somewhat, we will express them in terms of $n = \max(p, q)$. Observe that after one push any shape is in some stable equilibrium orientation. To simplify the exposition, we denote a stable equilibrium orientation of an object a *stable orientation* and a pair of stable equilibrium orientations of two objects a *stable pair*.

III. Enumerating realizable pairs of orientations

We are interested in finding push plans that bring P and Q into a fixed stable pair no matter what their initial orientation is. A stable pair for which this can be done is said to be *realizable*.

For the purpose of separating shapes some realizable stable pairs may be more interesting than others. For example, a pair that maximizes the separation of the centers of mass is more likely to be tolerant to faults or imprecisions. In this section we consider the problem of enumerating all realizable stable pairs, which allows finding the best separating pair for simple criteria.

A. Generalized push diagram

Let us first introduce the generalized push diagram, a tool to visualize the effect of a push operation on a pair of orientations. We partition the set $[0, 2\pi) \times [0, 2\pi)$ of all possible orientations of P and Q by vertical lines at every instable equilibrium of P and horizontal lines at every instable equilibrium of Q. Let $\mathcal{S}$ be the set of points of $[0, 2\pi) \times [0, 2\pi)$ corresponding to stable pairs. The couple $(\mathcal{I}, \mathcal{S})$ is the *generalized push diagram* (GPD) of the two parts P and Q (see Figure 3 for an example of such a generalized push diagram). There is exactly one point of $\mathcal{S}$ in each cell of $\mathcal{I}$.

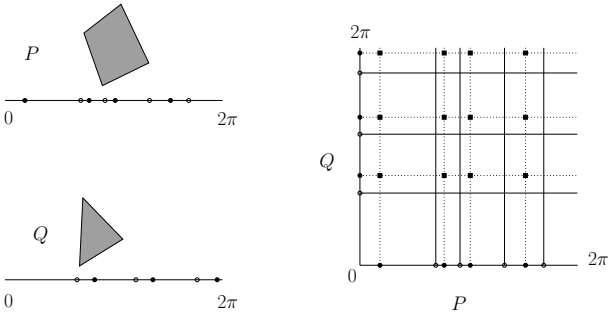

Fig. 3. Two parts, their respective push diagrams and the generalized push diagram of the pair.

Suppose that a push operation of angle β is applied to parts P and Q, initially in respective orientations ϕ and ψ. The resulting orientations of the shapes are the stable pair in the cell containing the point $(\phi+\beta, \psi+\beta)$. Thus, a push operation with angle β corresponds in the push diagram to a diagonal translation[1] of length $\beta\sqrt{2}$ followed by a snap-rounding operation to the stable pair of the cell (*c.f.* Figure 4). Since angles are represented modulo 2π the diagonal translation may wrap around, namely when $\phi + \beta > 2\pi$ or $\psi + \beta > 2\pi$.

B. Realizable pairs

In the GPD, the stable pairs reachable by one push from a specified pair of orientations (ϕ, ψ) are exactly those contained in cells intersected by the diagonal through (ϕ, ψ). The *pre-image* of a stable pair (ϕ, ψ) is the set of stable pairs that are mapped to (ϕ, ψ) by some push; it corresponds in the GPD to the set of stable pairs contained in the union of the diagonal lines intersecting the cell of (ϕ, ψ) (the shaded zone in Figure 5). Since a stable pair may have empty pre-image (as in Figure 5) it may not be realizable.

[1]Whenever we use the term *diagonal*, we mean the main diagonal direction (of slope 1). Thus a *diagonal translation* is a translation in direction of the vector $(1, 1)^T$, and the *diagonal through a point* is the line through that point of slope 1. Notice that a diagonal usually consists of two segments in the square $[0, 2\pi)^2$ due to the wrap around effect.

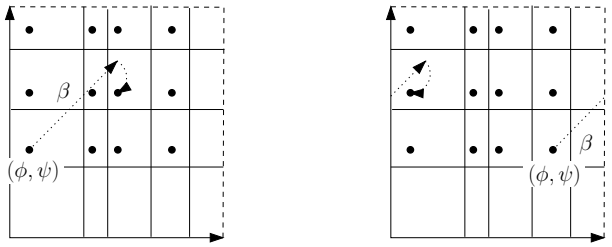

Fig. 4. In the generalized push diagrams, a push operation becomes a translation (possibly involving a wrap around as in the right example) followed by a snap-rounding.

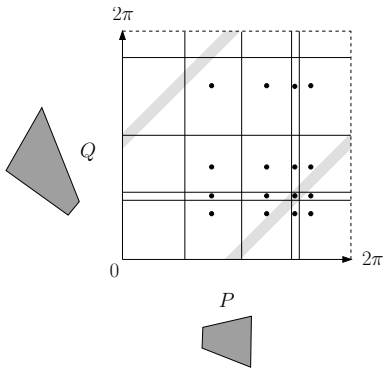

Fig. 5. The push diagram of a pair having a non-realizable pair of orientations.

However, there always exists one realizable pair and such a pair can be computed in time $O(n^2)$. Indeed, Goldberg [19] and Chen and Ierardi [15] proved that one shape can be oriented using $O(n)$ push operations that can be computed in $O(n^2)$ time. Once a part has been brought from any orientation into some fixed orientation, any additional push operation leaves it in some (possibly different) fixed orientation. We can thus compute a push plan orienting P and a push plan orienting Q in $O(n^2)$ time; the concatenation of these two push plans brings both P and Q into a stable pair irrespective of their initial orientations, yielding a realizable stable pair.

C. Enumerating realizable pairs

The *transition graph* is the directed graph whose nodes are stable pairs and whose edges connect a node to stable pairs it can reach by one push action.

Lemma 1: Let (ϕ, ψ) be a realizable stable pair. A pair (ϕ', ψ') is realizable if and only if it is reachable from (ϕ, ψ) in the transition graph.

Proof: Omitted. ■

The enumeration problem thus reduces to computing a realizable stable pair and performing a breadth-first search (BFS) in the transition graph. Computing a realizable stable pair can be done in time $O(n^2)$ as mentioned in Section III-B. The transition graph has size $O(n^3)$ since it has $O(n^2)$ nodes and each node has $O(n)$ out-going edges. Thus, using standard graph techniques it is possible to enumerate all realizable stable pairs in time $O(n^3)$. Taking advantage of the geometry of the problem, we can prune the transition graph more efficiently as we perform a BFS.

Theorem 2: The realizable pairs of orientations of two parts P and Q can be enumerated in $O(n^2 \log n)$ time and $O(n^2)$ space.

Proof: The basic step in the breadth-first search is this: given a node (which is in our case a stable pair, that is, a grid point in the push diagram), report all other nodes to which there is an out-going edge and that have not been visited before. The basic idea behind our algorithm is to build a data structure that allows us to quickly retrieve the nodes to which there is an outgoing edge. This data structure should support deletions so that we can delete a node from it as soon as it is reached for the first time. Hence, we will always only report nodes that have not been reached before.

Now consider a pair (ϕ, ψ) of stable orientations or, in other words, a grid point in the push diagram. We have seen before that there is an edge from (ϕ, ψ) to (ϕ', ψ') iff the diagonal through (ϕ, ψ) intersects the cell of(ϕ', ψ'). Hence, we need a data structure (allowing deletions) for the following queries: given a query line of slope 1, the diagonal through the point (ϕ, ψ), report all cells intersected by that line. Since the query line always have slope 1, we can project everything orthogonally onto a line with slope -1. The cells now become intervals, the query line becomes a point, and we wish to report all intervals containing the query point. This problem can be solved using an interval tree. An interval tree uses linear storage—in our case this is $O(n^2)$ since we have that many cells—and deletions take $O(\log n)$ time. Reporting all intervals containing a query point can be done in $O(\log n + A)$ time, where A is the number of reported intervals.

Since any interval is reported and deleted at most once, and the interval tree can be built in $O(n^2 \log n)$ time, the total time to do the BFS is $O(n^2 \log n)$. ∎

IV. Designing push plans

In this section we give a polynomial time algorithm to find an optimal sequence of push actions bringing any parts P and Q into some prescribed realizable stable pair, irrespective of their initial orientations. We first show how this problem reduces to computing a shortest path in some directed graph and then improve that computation by using the geometry of the problem.

An *antecedent* of a set of stable pairs A is any set of stable pairs B for which there exists a push action bringing any pair in B to some pair in A. The *antecedent graph* is the directed graph whose nodes are all sets of stable pairs and whose directed edges connect each node to all its antecedents.

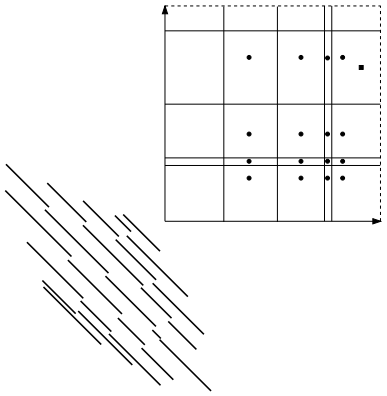

Fig. 6. The projections of the cells on the second diagonal.

A push plan that brings any shape P or Q into some prescribed realizable stable pair (ϕ, ψ) corresponds, in the antecedent graph, to a path from $A_1 = \{(\phi, \psi)\}$ to the set A_t of all stable pairs. Then, an optimal push plan is simply such a path with minimal length. The antecedent graph has order 2^{n^2} nodes which make it too large to be computed in practice.

The circular ordering on the angles induces a similar ordering on the stable orientations of a shape. Let a *circular interval* denote a set of stable orientations that are consecutive for this ordering (see Figure 7 for some examples). The product of two

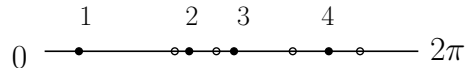

Fig. 7. Push diagram having for example $\{1, 2, 3\}$ or $\{3, 4, 1\}$ as circular intervals.

circular intervals corresponds, in the GPD, to a set of stable pairs whose cells make up a rectangle[2]. We call a product of two circular intervals a *circular rectangle*. The *reduced antecedent graph* is obtained from the antecedent graph by deleting all nodes that are not circular rectangles and their associated edges.

Lemma 3: The reduced antecedent graph contains $\{(\phi, \psi)\}$, the set of all orientations and at least one shortest path of the antecedent graph between them.

Proof: Obviously $\{(\phi, \psi)\}$ and the set of all orientations are circular rectangles. Let (A, B) be an edge in the antecedent graph with A being a circular rectangle. Let B' be the smallest circular rectangle that contains B. By monotonicity of the push function, any push action sending B in A also sends B' in A. Thus, B' is also an antecedent of A. Consider a shortest path $A_1 \to \ldots \to A_t$ from $A_1 = \{(\phi, \psi)\}$ to the set A_t of all orientations. By induction there exists a path $A_1 \to B_2 \to \ldots \to B_{t-1} \to A_t$ such that $A_i \subset B_i$ and the B_i are circular rectangles. This path is thus contained in the reduced antecedent graph and has the same length as a shortest path. ∎

[2]The wrap around effect can split it into four rectangles in the region $[0, 2\pi)^2$.

The reduced antecedent graph has $O(n^4)$ nodes, each having $O(n^2)$ out-going edges[3]. The size of the graph is thus $O(n^6)$ and standard graph techniques allows for a computation of an optimal push plan for a specified realizable pair in $O(n^6)$ time and space. Again, this result can be improved by using the GPD.

Consider a copy of a circular rectangle A sliding positively along the diagonal in the GPD. The circular rectangles that are antecedents of A correspond to the rectangles contained in that copy at the positions where it has at least one stable pair on its lower or left boundary. Therefore, there is an edge (A, B) in the reduced antecedent graph iff there exists a translation of B along the diagonal mapping the stable pairs of B to A with at least one being mapped to the lower or left boundary of A. We thus organize all the rectangles over the generalized push diagram in a data structure that allows for efficient answers to such queries.

Theorem 4: An optimal push plan achieving a specified realizable pair of orientations can be found in $O(n^4 \log^2 n)$ time and space.

Proof: Assume we are looking for rectangles B with at least one equilibrium point mapped to A's left boundary; the rectangles with an equilibrium point mapped to A's bottom side can be found with a similar data structure. It is not difficult to see that now the query translates to the following: find all rectangles B such that (a) the left side of B is contained in the left side of A when both sides are projected orthogonally onto a line of slope -1, and (b) the length of B's horizontal sides is less than or equal to the length of A's horizontal sides. Again we can solve this using a suitable geometric data structure. More precisely, the above query can be transformed to a 3-dimensional range query. Using a 3-dimensional range tree and dynamic fractional cascading (with only deletions) [28], we obtain a data structure using $O(s \log^2 n)$ storage and preprocessing, and with $O(\log^2 n)$ update and query time (plus, for queries, $O(1)$ time for each reported answer), where s is the number of objects stored in the structure. ■

V. Sorting k parts

Most of our results generalize to any number of polygonal shapes. Let $\{P_1, \ldots, P_k\}$ be a set of polygonal parts having $O(n)$ equilibrium points each. The problem now becomes to find a sequence of push actions that brings any shape into some fixed orientation such that the resulting positions of the centers of mass of the k shapes are pairwise separated. The two fundamental questions remain the design of algorithms to enumerate all realizable families of orientations, as some families may obviously not be realizable, and to compute an optimal push plan for a given realizable family.

Theorem 5: The realizable families of orientations of k polygonal parts $(P_1, \ldots, P_k)$ with $O(n)$ equilibrium points each can be enumerated in $O(n^k \log n)$ time.

[3]In fact, one can argue that the total number of edges in $O(n^5)$, but the running time of the algorithm we will present does not depend on the number of edges, so we do not go into the argument here.

Proof: By the same reasoning as in Section III-B a realizable family can be computed in time $O(kn^2)$. Generalizing the transition graph introduced in Section III-C to k parts is straightforward and the problem of enumerating all the realizable families also translates to a breadth-first search in some graph, which now has $O(n^k)$ nodes. The families of orientations accessible from a given family F correspond to the k-dimensional boxes whose projection on the hyperplane orthogonal to the main diagonal contain the projection of F. The dynamic data structure that supports the BFS now becomes somewhat more involved: it will be a multi-level data structure [3], whose first $k-2$ levels are segment trees and whose last level is an interval tree. Such a data structure on s boxes uses $O(s \log^{k-2} s)$ storage, $O(s \log^{k-1} s)$ preprocessing time, and queries and deletions can be done in $O(\log^{k-1} s)$ time (plus $O(A)$ time for reporting the answers in case of a query.) For lack of space, we omit the (standard) details. Since we have $s = O(n^k)$ cells, the running time of the entire algorithm is $O(n^k \log^{k-1} n)$. ■

Our algorithm to compute an optimal push plan generalizes in a similar way:

Theorem 6: An optimal push plan for a separating families of orientations of k polygonal parts $(P_1, \ldots, P_k)$ with $O(n)$ equilibrium points each can be found in $O(n^{2k} \log^{2k} n)$ time.

Proof: Omitted. ■

VI. Conclusion

We presented a polynomial-time algorithm to design a push plan for putting two types of parts into such orientations that they can be sorted using a simple sensorless device. Our approach generalizes to non-polygonal parts with finitely many equilibrium orientations, that is any shape whose boundary does not contain a circular arc centered at the center of mass.

Fence designs gained considerable attention recently as they can replace push mechanisms for certain tasks and are simpler to realize. We also plan to generalize our technique for sorting parts so that it works with fences instead of push actions. The main difference for our method is that the pushing angle of a fence is more restricted than that of a jaw. The diagonal query lines thus becomes diagonal query segments and our data structures over the generalized push diagram have to be adjusted accordingly.

An interesting open problem is to find the overall shortest sequence of push actions leading to any realizable stable pair of orientations, or to any realizable stable pair of orientations providing a prescribed minimum separation between the centers of mass of both parts. It would also be interesting to investigate the worst-case number of push actions needed to bring any two shapes into a given stable pair of orientations.

References

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

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

[3] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 2000.

[4] R-P. Berretty, K. Goldberg, M. H. Overmars, and A. F. van der Stappen. Algorithms for fence design. In *Robotics, the algorithmic perspective*, pages 279–295. A.K. Peters, 1998.

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

[6] R-P. Berretty, K. Goldberg, M. H. Overmars, and A. F. van der Stappen. Orienting parts by inside-out pulling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1053–1058, 2001.

[7] R-P. Berretty, M. H. Overmars, and A. F. van der Stappen. Orienting polyhedral parts by pushing. *Computational Geometry: Theory and Applications*, 21:21–38, 2002.

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

[9] K-F. Böhringer, B. R. Donald, and N.C. MacDonald. Upper and lower bounds for programmable vector fields with applications to mems and vibratory plate part feeders. *Algorithms for Robotic Motion and Manipulation*, J.-P. Laumond and M. Overmars (Eds.), A.K. Peters, pages 255–276, 1996.

[10] M. Brokowski, M. A. Peshkin, and K. Goldberg. Optimal curved fences for part alignment on a belt. *ASME Transactions of Mechanical Design*, 117, 1995.

[11] R. Brost. Automatic grasp planning in presence of uncertainty. *International Journal of Robotics Research*, 7(1):3–17, 1988.

[12] R. C. Brost and K. Goldberg. A complete algorithm for synthesizing modular fixtures for polygonal parts. *IEEE Transactions on Robotics and Automation*, 12:31–46, 1996.

[13] J. Canny and K. Goldberg. A risc approach to robotics. *IEEE Robotics and Automation Magazine*, 1:26–28, 1994.

[14] J. Canny and K. Goldberg. Risc for industrial robotics: Recent results and open problems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1951–1958, 1994.

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

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

[17] M. A. Erdmann, M. T. Mason, and G. Vaněček. Mechanical parts orienting: The case of a polyhedron on a table. *Algorithmica*, 10(2):226–247, 1993.

[18] K. Goldberg. Handling industrial parts with the parallel-jaw gripper. In *NSF Design and Manufacturing Systems Conference*, 1993.

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

[20] K. Goldberg, B. Mirtich, Y. Zhuang, J. Craig, B. Carlisle, and J. Canny. Part pose statistics: Estimators and experiments. *IEEE Transactions on Robotics and Automation*, pages 849–859, 1999.

[21] W. Huang and M. T. Mason. Mechanics, planning and control for tapping. In *Robotics, the algorithmic perspective*, pages 91–102. A.K. Peters, 1998.

[22] D. Kriegman. Let them fall where they may: Capture regions of curved objects and polyhedra. *International Journal of Robotics Research*, 16:448–472, 1997.

[23] K. M. Lynch. Locally controllable manipulation by stable pushing. *IEEE Transactions on Robotics and Automation*, pages 314–323, 1999.

[24] K. M. Lynch. Toppling manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2551–2557, 1999.

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

[26] A. Marigo, M. Ceccarelli, S. Piccinocchi, and A. Bicchi. Planning motions of polyhedral parts by rolling. *Algorithmica*, 26(4):560–576, 2000.

[27] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[28] K. Mehlhorn and S. Näher. Dynamic fractional cascading. *Algorithmica*, 5:215–241, 1990.

[29] Mark Moll and Michael A. Erdmann. Manipulation of pose distributions. *International Journal of Robotics Research*, 21(3):277–292, 2002.

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

[31] M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, pages 696–701, 1988.

[32] A. Rao and K. Goldberg. Shape from diameter: Recognizing polygonal parts with a parallel-jaw gripper. *International Journal of Robotics Research*, 13(1):16–37, 1994.

[33] A. Rao and K. Goldberg. Friction and part curvature in parallel-jaw grasping. *Journal of Robotic Systems*, 12(6):365–382, 1995.

[34] A. Rao and K. Goldberg. Manipulating algebraic parts in the plane. *IEEE Transactions on Robotics and Automation*, 11:589–602, 1995.

[35] A. Rao, D. Kriegman, and K. Goldberg. Complete algorithms for reorienting polyhedral parts using a pivoting gripper. *IEEE Transactions on Robotics and Automation*, 12(2):331–342, 1996.

[36] A.S. Rao and K. Goldberg. Placing registration marks. *IEEE Transactions on Industrial Electronics*, 41:51–59, 1994.

[37] D. Reznik and J. Canny. Universal part manipulation in the plane with a single horizontally-vibrating plate. In *Robotics, the algorithmic perspective*, pages 23–34. A.K. Peters, 1998.

[38] A. Sudsang, J. Ponce, and N. Srinivasa. Algorithms for constructing immobilizing fixtures and grasps of three-dimensional objects. In J-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 363–380. A.K. Peters, 1997.

[39] R. Wagner, Y. Zhuang, and K. Goldberg. Fixturing parts with seven modular struts. In *IEEE International Symposium on Assembly and Task Planning*, pages 133–139, 1995.

[40] A. S. Wallack and J. Canny. Planning for modular and hybrid fixtures. *Algorithmica*, 19(1–2):40–60, 1997.

[41] M.Y. Wang. An optimum design for 3d fixture synthesis in a point set domain. *IEEE Transactions on Robotics and Automation*, 16:839–846, 2000.

[42] M.Y. Wang and D. Pelinescu. Optimizing fixture layout in a point set domain. *IEEE Transactions on Robotics and Automation*, 17:312–323, 2001.

[43] C. Wentink, A. F. van der Stappen, and M. H. Overmars. Algorithms for fixture design. In J-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 321–346. A.K. Peters, 1997.

[44] D. E. Whitney. Real robots don't need jigs. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 746–752, 1986.

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

[46] T. Zhang, G. Smith, R-P. Berretty, M. H. Overmars, and K. Goldberg. The toppling graph: Designing pin sequences for part feeding. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 139–146, 2000.

[47] Y. Zhuang and K. Goldberg. On the existence of solutions in modular fixturing. *International Journal of Robotics Research*, 15:646–656, 1996.