The Automated Design of Parts Orienters

B.K. Natarajan

TR 86-774
August 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853

THE AUTOMATED DESIGN OF PARTS ORIENTERS

B.K. Natarajan
Cornell University
Ithaca, NY 14850

This paper concerns the design of parts orienters – the dual to the motion planning problem. Three particular paradigms are considered and their abstractions to the computational domain lead to interesting problems in graph pebbling and function composition on finite sets. Polynomial time algorithms are developed for the abstracted problems.

1. INTRODUCTION

1.1 Motivation

Robots are seeing increasing use in assembly tasks. The typical assembly task of mating two parts follows the following paradigm; the robot picks up the first component and positions it at a convenient location. It then picks up the second component and mates it with the first. The problem is that today's robot cannot adjust well to variations in the position and orientation of the part when it attempts to grasp it. Consequently, it is necessary to feed the part to the robot in specific orientations. Typically, this requires a parts orienter, which is a 'black box' that accepts the part in any orientation and outputs it in some predetermined orientation. Currently, parts orienters are designed by the 'paper and pencil' method and may require many hours of work. In the CAD/CAM environment, where the design of a simple part can be completed in a matter of minutes, it is desirable that the complete assembly process for the part be determined in about the same time. Hence, there is a need to automate the design of parts orienters for simple parts and integrate the design algorithm with the CAD/CAM system. What would such an algorithm look like? It should take as inputs

(a) Geometric information on the part, such as is typically resident in a CAD/CAM database,

(b) The set of orientations in which the part may enter the orienter and the set of orientations in which it should exit,

and should output a suitable feeder for the part, if such exists within the design framework. If none exists, the algorithm should perhaps suggest modifications to the part which would permit such a feeder. Hopefully, such modifications will not alter the functional nature of the part.

The above problem is one of real-world interest and in many ways is similar to the problem of building compilers for programming languages faced in the fifties and the sixties. Then, little was known about the theory of compilers or context free grammars and the first compilers were hand-crafted around the language. With the development of a theory, it was possible to identify the class of languages that permitted simple and elegant compilers. Today, it is straight forward to generate compilers automatically for this class, eliminating many hours of hard labour. There is reason to believe that a similar situation exists in robotics and that it is necessary to build abstractions and theory for what are generally deemed as 'hard but gritty' real world problems. For example, it would be useful to identify the class of parts for which the design of orienters can be easily automated.

This paper attempts to abstract and analyze the design of parts orienters. The abstracted problems are not only interesting from a theoretical viewpoint, but also provide useful insight into the physical domain.

1.2 The Problem Statement

It is elegant and intuitively helpful to define the problem of parts orienter design as follows:

Given an object O and two sets of orientations I and G for it, design a set of obstacles S that forces the object to move from any orientation in I to some orientation in G.

This set of obstacles S is the 'black box' orienter that will accept O in any orientation in I and output it in some orientation in G. Notice that this definition is in some sense the inverse of the motion planning problem:

Given an object O and two sets of configurations I and G for it in a scene S of obstacles, are there paths for the object from every configuration in I to some configuration in G?

The difficulty with the above definition of the parts orienter design problem is that it does not completely capture the physics involved. If the obstacles to be designed are static, how can they force the part to move? Tacit here is some external force that drives the part and without being precise about the exact nature of this driving force, it is impossible to proceed. Hence, it is necessary to study different paradigms for parts orienters and attempt to devise automating algorithms for each of these paradigms. Towards this end, we classify parts orienters as they are in practice, [Boothroyd et al, 1982, Philip, 1980].

1.3 Classifying Parts Orienters

Fig. 1 shows a heirarchical classification of parts orienters. The root of the

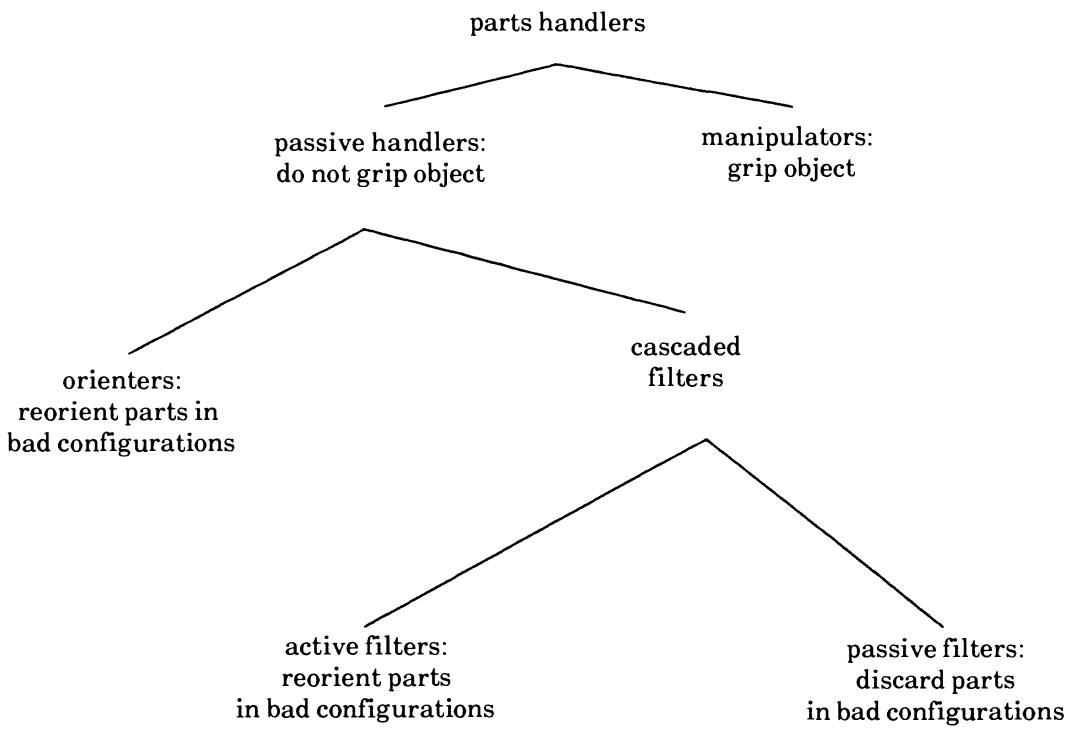

Fig. 1. Classifying parts orienters

tree is labelled *parts handlers* and includes any and all devices that may be used to manipulate parts. This general class is subdivided into *passive handlers* and *manipulators*. Manipulators are devices that explicitly grip the object that they handle. Typical examples of this class are robots and remote handlers for toxics.

Passive handlers are devices that do not grip the object they handle. Examples of this class are conveyor belts, vibratory feeders, etc. The distinction between the two classes is not rigorous as one could well consider devices that reduce the number of degrees of freedom of the object handled but do not constrain them entirely.

In this paper we are primarily interested in passive handlers as "preprocessors" to the much studied class of manipulators, [Udupa, 1977, Lozano-Perez and Wesley, 1979, Reif, 1979, Schwartz and Sharir, 1982]. Passive handlers can be further subdivided into *orienters* and *cascaded filters*. Orienters are best described as devices that accept the object to be handled in any orientation and output it in the desired orientation. Cascaded filters are, as the name suggests, a sequence of filters. A filter is a three port device that accepts the object in some set of orientations C_i and delivers it in another set of orientations C_o, possibly discarding the object if it enters the filter in some set of orientations C_d. Fig. 2 is a schematic of a filter. C_i will in general not be the set

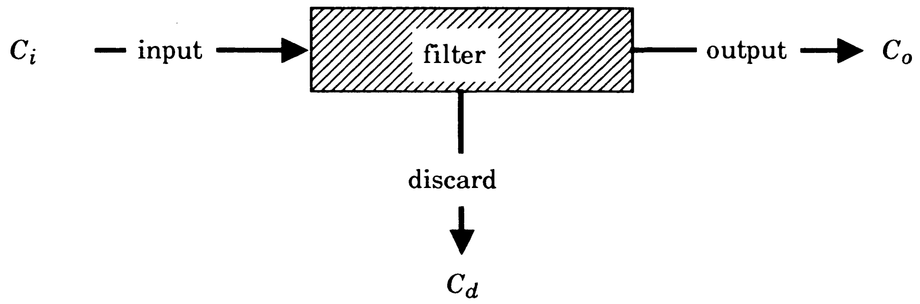

Fig. 2. Schematic of a filter

of all orietations C_u and if the object is fed to the filter in a orientation in C_u–C_i the filter gets 'stuck' and its effect on the part is undefined. If the object traversing the filter starting in every orientation in C_i is either discarded or allowed to exit the filter without reorientation, then the filter is a *passive filter*. Otherwise, the filter is an *active filter*. Typically, cascaded filters are configured as shown in Fig. 3. Parts are fed to the filter sequence from a randomizing bin so that they occur in all configurations and the discarded parts from each filter are

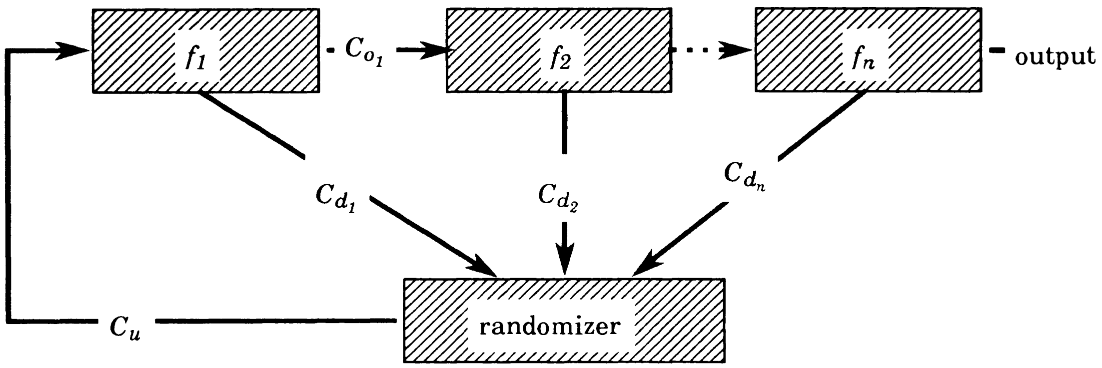

Fig. 3. Schematic of a cascaded filter

returned to the bin. Most vibratory feeders follow this paradigm, [Boothroyd et al, 1982].

The main difference between orienters and cascaded filters is that filters enjoy the additional freedom of feedback to recycle parts that end up in bad orientations. The distinction is not rigorous as, for example, a cascaded filter consisting entirely of active filters that do not discard any orientations ($C_d = \phi$) is also an orienter. Cascaded filters are most useful in orienting small parts that are to be handled in large quantities, while orienters are suitable for few-of-a-kind parts, large parts, and task driven situations like those that a household robot might face.

1.4. Overview of Results

Having classified parts orienters in a general way, we proceed by studying particular paradigms for the two classes of passive handlers - orienters and filters. In the orienters class, we consider two paradigms - the belt and the panhandler. The belt is for two dimensional polygonal objects that are infinite in the third dimension while the panhandler is for flat polygonal objects. In the cascaded filters class, we show how to automate the design of vibratory feeders for flat, polygonal objects. In each case, the physical problem is first made precise by making suitable assumptions and then abstracted to a computational problem. The abstracted problems bring up interesting questions in graph pebbling and selective function composition on finite sets. We restrict our attention to polynomial time algorithms to these problems without regard to exponents or constants.

2. PARADIGMS FOR ORIENTERS

2.1 The Belt

Consider a two dimensional belt (infinite in the third dimension) as shown in Fig. 4. The belt moves from left to right and has vertical jumps at various points.

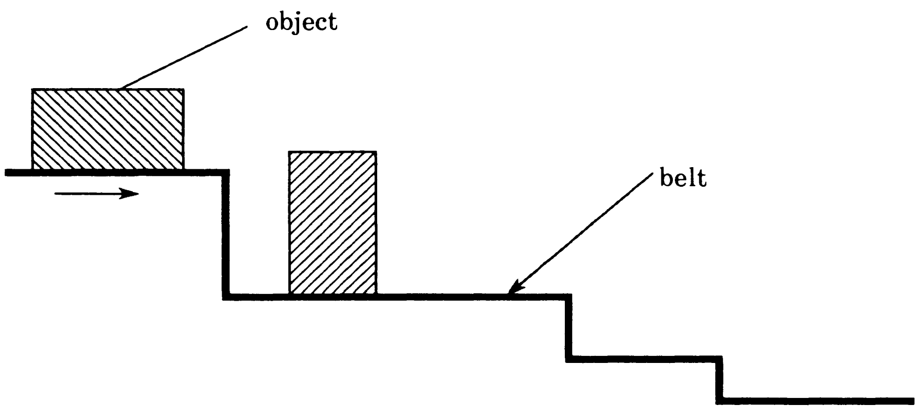

Fig. 4. The belt paradigm

The two dimensional convex polygonal object (infinite in the third dimension) is placed on the left end of the belt in some orientation. As the object traverses a jump, it could be reoriented depending on the height of the jump and the orientation in which the object approached the jump. The aim is to pick a sequence of jumps so that regardless of the orientation in which the object starts on the belt, it always attains the same predetermined orientation at the end of the belt. This paradigm could be useful for orienting large objects like say, cartons that are to be oriented before being unpacked by a robot.

The Problem Statement and Abstraction

We consider an n-sided convex polygonal object O and label the faces 1 through n. The object is in orientation i if it lies on face i. Referring to Fig. 5, we define the *transition height* h_i of orientation i to be the minimum jump height at

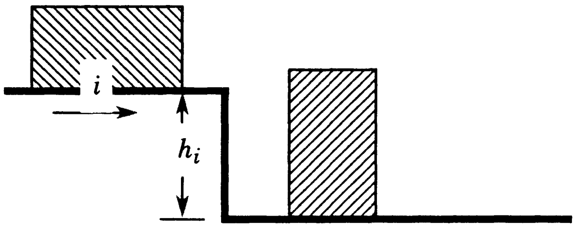

Fig. 5. The transition height h_i

which a change in orientation occurs when the object aproaches the jump in orientation i. We assume that the transitions in the above definition result in unique orientations. Furthermore, if the object in orientation i is subject to a jump of height much greater than h_i, it will tend to tumble before attaining a stable configuration. Since such behaviour is unpredictable and inconsistent, we will disallow such possibilities by placing the restriction that if the object can be in orientation i, it cannot be subject to a jump greater than h_i. We are now ready to state the problem.

Problem 1

Given a convex, n-sided polygonal, two dimensional object and the transition height for each orientation of the object, find a sequence of jumps so that the object started on the belt in any orientation always attains some predetermined orientation at the end of the belt.

We assume that the transition heights of the object are given as they are too strongly dependent on the physical parameters of the problem to be within the scope of this paper. They might be determined by n simple experiments or computations. Define the transition diagram of an object to be a graph with one vertex per orientation of the object. An edge (u,v) with label t is present in the diagram if and only if orientation u when subject to its transition height of t goes to orientation v. Since the transitions are unique, the graph is a collection of cycles with zero or more trees attached by the root to each cycle. We can now state the abstract version of the problem as a graph pebbling game on the transition diagram.

Problem 1'

Pebbling Game

start : All vertices are pebbled.

move: Of all pebbled vertices, pick those with minimum labels on their respective outgoing edges. Move the pebbles along these edges.

Question

Find a sequence of moves that minimizes the number of pebbled vertices.

Fig. 6 is a simple algorithm to solve the problem. Essentially, the algorithm

```
Algorithm 1
    S ← {1,2,...n};        /* set of pebbled vertices */
    σ ← ε;             /* move sequence, initially null */
    for i ← 1,2,...,m do
        h ← min{h_k|k is in S}; /* find minimum label on edges out of pebbled vertices */
        S ← h(S);            /* move pebbles along minimum label edges */
        σ ← σ∘h;             /* update move sequence     */
    od
```

Fig. 6. The pebbling algorithm

considers the set of pebbled vertices of the graph and plays the pebbling game for a predetermined number of moves. When the algorithm terminates, the set of pebbled vertices is of minimum size in the sense that no other sequence of jumps produces a smaller set. The interesting question here is how big the sequence length m need be.

Claim 1: The sequence length m is $\Theta(n^2)$.

Proof: We first show that m can attain n^2.

Subclaim: The sequence length m attains n^2.

Proof: Consider the transition diagram of Fig. 7. It is clear that any sequence of jumps producing the minimum set of pebbled vertices for this diagram is of length at least $1+2+3+4+5+6 = n(n+1)/2$ for $n=6$. Since there exists such a diagram for every n, we conclude that m is $\Omega(n^2)$. Note

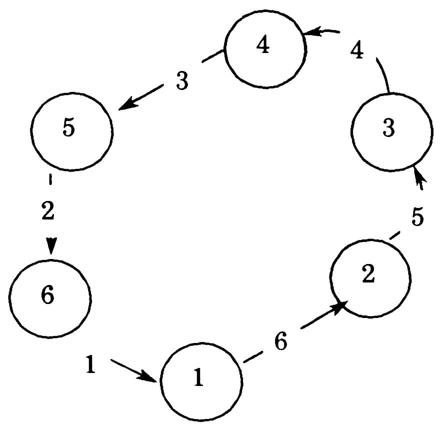

Fig. 7. Worst case transition diagram

that we have said nothing about the objects that generates this family of diagrams and cannot do so as as we do not have sufficient knowledge of the physics. □

Next we show that m is $O(n^2)$. We say that an edge (u,v) with label t of the transition diagram is *used* by a move if u was selected by the move (i.e, before the move u was a pebbled vertex and the label value t was a minimum).

Subclaim: Every edge in the transition diagram is used $O(n)$ times when the minimum set of pebbled vertices is attained.

Proof: There are two kinds of edges in a transition diagram. Those that are in a cycle and those that are not. Let (u,v) be an edge that is not on a cycle. Clearly, (u,v) can be used only as many times as there are vertices z such that there exists a path from z to u. Since there can be at most n such, the claim follows.

Now let (u,v) be an edge on a cycle. We work the proof in stages. Suppose the transition diagram consists entirely of a single cycle. It is not hard to see that when any edge of the cycle is used n times, every edge of the cycle must already have been used, including those edges on the cycle with maximum label value. Consider the set of edges on the cycle with maximum label value. It is straightforward to verify that the first time any of these edges are used, all of them will be used and that the minimum set of pebbled vertices is attained. Since the set of pebbled vertices can never grow, any further moves will maintain the minimality of the set.

Now extend the diagram to consist of a single cycle with some trees attached to it. Let t_m be the maximum label value of the edges on the cycle and let $U = \{(u_i,v_i)\}$ be the edges on the cycle with this label value. Also, let $R = \{(r_i,s_i)\}$ be the tree edges with a label value of t_m. Notice that either every edge in U is used at any move or none of them are used, and that if any edge of R is used then every edge of U is used. Also, edges with label value greater than t_m will never be used. Now, the first time an edge from U is used, the set of pebbled vertices is

$\{u \mid (u,v)$ has label $\geq t_m\}$.

Furthermore, no edge in the diagram has been used more than n times. Call a tree *active* if the set

$\{u \mid u$ is a pebbled vertex in the tree and the edge out of u has label $\leq t_m\}$

is nonempty. Notice that trees that are not active cannot be reduced further. Between two succesive uses of edges in U, the number of pebbled vertices on each active tree decreases by one and the cycle edges between two edges (u_1,v_1) and (u_2,v_2) in U (as shown in Fig. 8) are each used no more than once

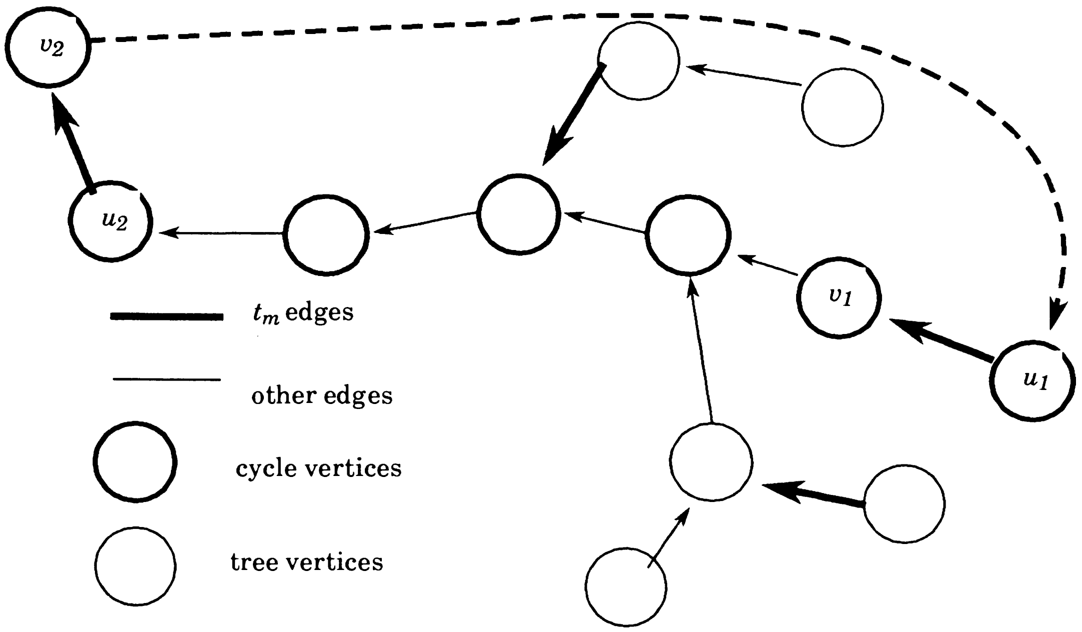

Fig. 8. The segment between cycle edges of maximum label value t_m.

for each active tree attached to the segment $u_1,v_1,\ldots,u_2,v_2$. Hence after $O(n)$ uses of the cycle edges, all the trees will be inactive and the minimim set of pebbled vertices attained. This implies that no edge is used more than $2n$ times before the minimum set of pebbled vertices is attained.

Now consider the transition diagram with multiple cycles and trees. Let

$$t_m = \min_{C_i \text{ is a cycle}} \; \max_{(u,v) \text{ in } C_i} \{\text{label}(u,v)\}.$$

Clearly, t_m is the maximium edge label that can be used. Pick any cycle in the diagram with an edge labelled t_m. Now the arguments made above apply to this cycle. The first time an edge labelled t_m is used, no edge could have been be used more than n times. After that, each edge on the cycle is used once for each vertex in an attached tree on this or some other cycle of the diagram.

The two subclaims completes our proof. □

With the above discussion, we can tighten the loop guards of Algorithm 1 to obtain Algorithm 1' of Fig. 9.

Algorithm 1'

$t_m \leftarrow \min_{C_i \text{ is a cycle}} \; \max_{(u,v) \text{ in } C_i} \{\text{label}(u,v)\}$;

$\sigma \leftarrow \varepsilon$; /* move sequence, initially null */

$S_m \leftarrow \{u|(u,v)$ is a cycle edge with label $t_m\} \cup$

$\{u|(u,v)$ is a tree edge with label $\geq t_m\}$; /* minimal set of pebbled vertices */

$S \leftarrow \{1,2,...n\}$; /* set of pebbled vertices */

while $|S| \neq |S_m|$ **do**

$h \leftarrow \min\{h_k|k$ is in $S\}$;

$S \leftarrow h(S)$; /* move pebbles along minimum label edges */

$\sigma \leftarrow \sigma \circ h$; /* update move sequence */

od

Fig. 9. An improved pebbling algorithm

Remarks

We considered unidirectional belts in the above treatment. It might be of interest to consider bidirectional belts, where the object could be tranferred from a belt moving right to a belt moving left and vice versa. See Fig. 10. Although,

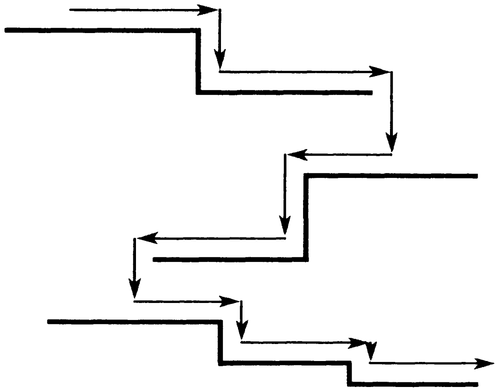

Fig. 10. A bidirectional belt.

this does not change the worst case jump sequence length, it might be significantly better in some cases.

An average case analysis of the algorithm might prove it to be better than $O(n^2)$. In itself, an $O(n^2)$ bound is not very encouraging.

We assumed that the transitions at any jump were single valued. It is more realistic and perhaps more interesting to relax this restriction.

2.2 The Pan Handler

Here we consider the problem of orienting planar rigid polygonal objects in task driven situations. i.e. the same orienter is to be used to handle a vareiety of such parts. The orienter consists of a planar regular polygonal tray with a lip as shown in Fig. 11. Suppose the object is dropped onto the tray and is in some

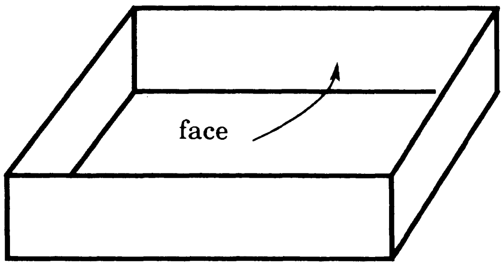

Fig. 11. The orienting tray

possible set of orientations against a certain face of the tray. If the tray is tilted, the object leaves this face and strikes another face depending on the tilt direction, possibly reorienting itself when it attains a stable configuration against the new face. The problem is to choose a sequence of tilt directions that will reorient the object in some unique orientation against a predetermined face of the tray, if such a sequence exists. This scheme was first proposed in [Grossman and Blasgen, 1975] and more recently discussed in [Mason and Erdmann, 1986]. The paradigm might be useful for household robots that could walk around with trays so that when one drops say, a polygonal coffee mug in the tray, the robot walks off tilting the tray until it has uniquely oriented the mug and picked it up.

Problem Statement and Abstraction

Before proceeding with the problem we need to make some assumptions and place some restrictions to state the problem more precisely. Our assumptions are:

(a) The object is allowed to contact only one face of the tray at a time.

(b) One and only one edge of the object makes contact with any face of the tray in any orientation. To support this, we assume the object to be convex. This is without loss of generality as we can always consider the convex hull of a non-convex object.

(c) The object always remains roughly in the middle of a face at any time. In light of assumption (e), this is equivalent to the tray being large compared to the object.

(d) All operations are planar in that the object is always dropped on a distinguished face of its two planar faces and it lies on that face throughout.

(e) The tray can only be tilted along median directions - lines joining midpoints of two faces as shown in Fig. 12. This is to discretize the set of tilt directions for two reasons: A fixed and finite set of tilt directions does not require a full freedom wrist to tilt the tray but can be achieved with a fixed and finite set of binary actuators. Secondly, as we shall show later, the problem is not as interesting if any tilt direction is attainable.

(f) The faces of the tray have a high coefficient of friction while the tray floor is smooth. This allows the object to rotate about the vertex of first contact without sliding when the tray is tilted. See Fig. 13.

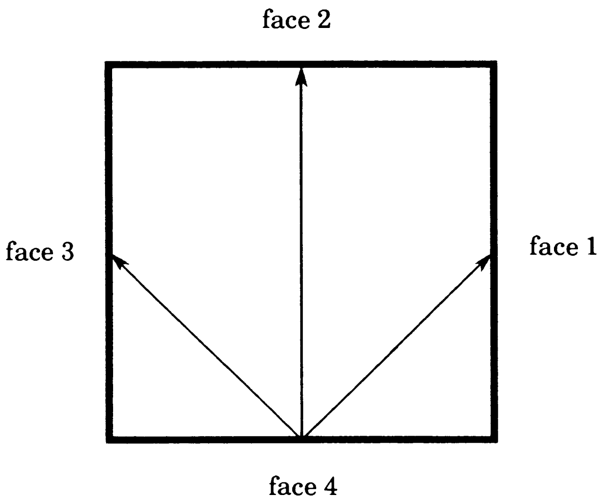

Fig. 12. Median directions

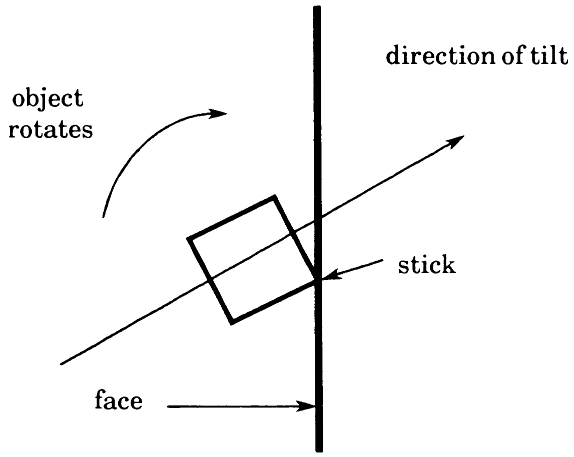

Fig. 13. Rotation about the vertex of first contact

(g) The tray is never tilted so steeply that the object tumbles, i.e. rotates past its first stable configuration when it makes contact with a face, as shown in Fig. 13.

(h) The transitions are single valued in that if the object is in a certain orientation and the tray is tilted in a median direction, the resulting orientation is uniquely known.

(h) Initially, the object can be in any allowable configuration against some predetermined face.

We are now ready to state the problem:

Problem 2

Given a k-sided regular polygonal tray and a n-sided convex planar polygonal object, find a sequence of tilt directions that reduces the possible orientations to any single orientation under the above assumptions, if there exists such.

Since the tray is regular, we need not distinguish between the object lying in the same orientation against two different faces. Also, the effect of each tilt direction is independent of the face to within an isomorphism. We number the faces in say anti-clockwise order from any face so as to give the median directions with respect to that face a unique labelling. Fig. 12 shows the labels for the median directions with respect to face 4. Hereafter, we refer to the median directions by these labels so that the j^{th} direction corresponds to the j^{th} face in this ordering. Labelling the sides of the object in cyclic order (say anti-clockwise), we say that the object is in orientation i if it lies on side i. Corresponding to this labelling, we define S to be the set of n orientations of the object. We can now express the effect of median tilts as k functions $f_1, f_2, \ldots, f_k$ from S to S and state the abstract version of the problem in terms of these mappings.

Problem 2'

Given k mappings $f_1, f_2, \ldots, f_k : S \rightarrow S$ on a finite set S, find a mapping f_0 that is a composite of the f_i's such that f_0 is a constant function on S. i.e.,
$|f_0(S)| = |\{s | s = f_0(v) \text{ for some } v \text{ in } S\}| = 1$.

Fig. 14 gives an algorithm to solve the abstracted problem above. The algorithm is greedy and makes no attempt to be efficient or provide an optimal solution. Starting with the identity function, at each stage the algorithm attempts to find a composite function with range at least one less than the previous stage. To do this, it picks a pair of elements from the range of the previous stage, and looks for a sequence of function compositions that merges these two. It then composes this sequence with the function of the previous stage to obtain the new function. If no such pair of elements exists it halts and reports failure.

The algorithm works because the problem satisfies the inclusion property: if there exists $s \in S$ and composite function $f_0 : S \rightarrow s$, then for any subset $S' \subseteq S$ there

Algorithm 2

(1) determine the transitions for each face and each median direction.

$S \leftarrow \{1,2,...n\}$; /* possible orientations */

$f_0 \leftarrow \varepsilon$; /* identity mapping */

While $|S|>1$ **do**

(2) pick i and j in S such that there exists a sequence of function compositions s such that $s(i) = s(j)$. If no such exists, halt and report failure.

(3) $S \leftarrow s(S)$; /* update S */

$f_0 \leftarrow s \circ f_0$; /* update composite sequence */

od

Fig. 14. The function composition algorithm

exists a composite function f such that $f(S')=s$ (f_0 is trivially such a function). Consequently, the greedy algorithm can reduce the range set whenever and however it can, without ever getting into some set of elements that cannot be reduced, although there might have been some other reducing sequence.

Time Analysis

Step (1) can be computed in $O(nk)$ time or by $O(nk)$ simple experiments.

Step (2) can be computed in $O(kn^2)$ time. To see this, notice that we can construct a transition diagram for pairs of elements in S. Specifically, this is a directed graph $G=(V,E)$ where

$V=\{(i,j) \mid i,j \in S\}$

$E = \{(i_1,j_1)\text{->}(i_2,j_2) \mid$ if $f_i(i_1)=i_2$ and $f_i(j_1)=j_2$ for some $1 \le i \le k\}$

label$((i_1,j_1)\text{->}(i_2,j_2)) = i$, if $f_i(i_1)=i_2$ and $f_i(j_1)=j_2$ for some $1 \le i \le k$.

Every vertex has outdegree k and there are n^2 vertices and hence at most n^2k edges. Now, a depth-first search of this graph yields a path from a vertex (i_1,j_1) to (i_2,j_2) where $i_1 \neq j_1$and $i_2=j_2$. The labels on this path form the composition sequence s. The depth-first search requires $O(kn^2)$ time and so does constructing the graph, although this need be done only once.

Step(3) takes $O(kn^3)$ as it requires applying the $O(kn^2)$ sequence to the set S which is of size $O(n)$.

Since the algorithm can run for at most n iterations, the total running time is $O(kn^4)$ and the length of the sequence generated by the algorithm is $O(kn^3)$.

A Modified Problem

The above, assumed that the object can be in any orientation against a particular face at the start of the tilt sequence (assumption (h)). Since all orientations of the object may not be simultaneously stable, this assumption is a bit too strong. In particular, consider the case where the object starts in some given set of orientations against a particular face. Although this is a relatively simple extension, the *abstract* problem as stated in Problem 2' becomes PSPACE-complete. We shall not prove this claim here, but cite the related results of [Kozen, 1977], where the problem of non-empty intersection of regular sets is proved PSPACE-complete. Intuitively, it seems rather unlikely that the physical problem also exhibits such a jump in complexity, leading us to believe that the abstraction of Problem 2 to Problem 2' is not tight enough.

Consider the object lying on side 1 as shown in Fig.15. Suppose that tilting in

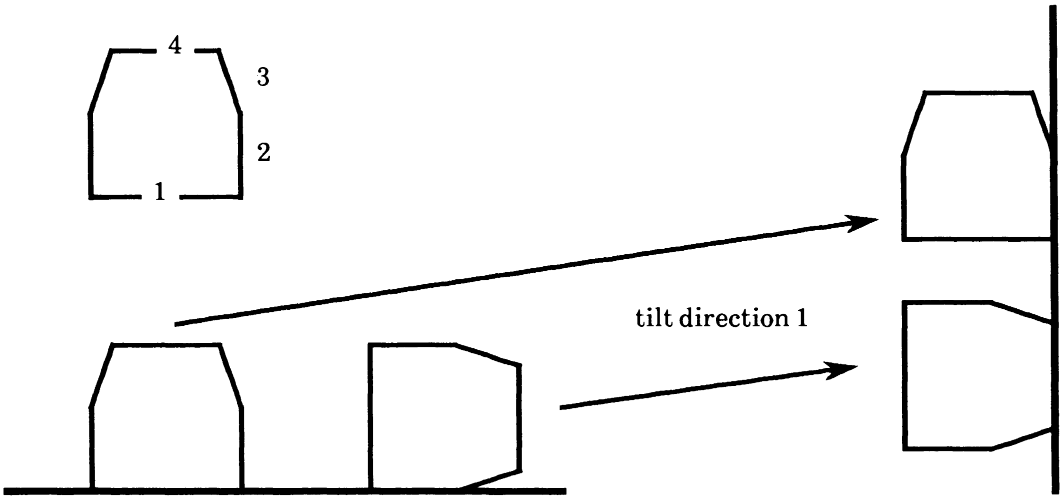

Fig. 15. Monotonicity of tilt functions

direction 1 leaves the object lying on side 2. Then, if the object had started on side 2, it must end up "at least" (in anti-clockwise order) on side 2 after the tilt. In short, the tilt functions $f_1, f_2, \ldots, f_k$ must be *monotonic* with respect to the cyclic order on the set of orientations S. To make this more precise, notice that any oriented simple cycle that contains every element of a finite set generates a cyclic order on the set. A sequence $s_1,s_2,\ldots,s_j$ of elements of the set is *ordered* if $s_1,s_2,\ldots,s_j$ are encountered in that order when the generating cycle of S is traced <u>once</u>, starting from s_1. A function $f:S\rightarrow S$ is monotonic if for any ordered sequence $s_1,s_2,\ldots,s_j$, the sequence $f(s_1), f(s_2),\ldots,f(s_n)$ is ordered. Now, recalling that S has a cyclic order on it generated by the anti-clockwise numbering of the sides of the object, we are ready to state the abstracted problem afresh.

Problem 2"

Given k monotonic functions $f_1, f_2,\ldots, f_k :S\rightarrow S$ on a finite set S with a cyclic order, and a subset U of S, find a function f_0 that is a composite of the f_i's such that $|f_0(U)| = 1$.

Before we can give an algorithm to solve the above problem, we need some supporting lemmas and definitions. Let S be a finite set with a cyclic order and let $\sigma=s_1,s_2,\ldots,s_n$ be an ordered enumeration of S. The interval $[s_i,s_j]$ of σ is the set $\{s_i,s_{i+1},\ldots,s_j\}$ if s_i precedes s_j in σ. Otherwise, $[s_i,s_j] = [s_i,s_n]\cup[s_1,s_j]$. Let U be a subset of S. We say U *partitions* σ into the intervals $[u_1,u_2]$, $[u_2,u_3]$,...$[u_j,u_1]$ of σ if $u_1,u_2,\ldots u_j$ is an ordered enumeration for U such that u_1 appears before u_j in σ.Notice that for a given cyclic order, the partition is unique.

Lemma 1: Let S be a finite set with a cyclic order and let $\sigma=s_1,s_2,\ldots,s_n$ be an ordered enumeration for it. Also, let $f:S\rightarrow S$ be a monotonic function such that for some s_i and s_j in S, $f(s_i)=f(s_j)=s$. Then, if $\{s_i, s_j\}$ partitions σ into $[s_i,s_j]$ and $[s_j,s_i]$, either $f([s_i,s_j])=s$ or $f([s_j,s_i])=s$.

Proof: Immediate from the monotonicity of the function f. □

Lemma 2: Let S be a finite set with a cyclic order and let $\sigma=s_1,s_2,\ldots,s_n$ be an ordered enumeration for it. Let $f:S\rightarrow S$ be a monotonic function such that $|f(S)|>1$, and let U be a proper subset of S partitioning σ into intervals $[u_1,u_2]$, $[u_2,u_3]$,...$[u_j,u_1]$. (Let $u_{j+1}=u_1$). Then, $f(U)$ is a singleton set $\{s\}$ if and only if one of the intervals $[u_i,u_{i+1}]$ is such that

(a) $f(u_i)=f(u_{i+1})=s$

(b) for some $v\in [u_i,u_{i+1}]$, $f(v) \neq s$.

Proof:

(only if)

There exists v in S such that $f(v) \neq s$. Now v belongs to some interval $[u_i,u_{i+1}]$ that satisfies condition (b). Since $f(U)=\{s\}$, $f(u_i)=f(u_{i+1})=s$ satisfying condition (a). Hence the claim is proved.

(if)

Let v in $[u_i,u_{i+1}]$ be such that $f(v) \neq s$. Since f is monotonic, we can apply Lemma 1 to the partitioning of S by $\{u_i,u_{i+1}\}$ and conclude that either $f([u_i,u_{i+1}])=\{s\}$ or $f([u_{i+1},u_i])=\{s\}$. But $f([u_i,u_{i+1}])\neq\{s\}$ as $v\in[u_i,u_{i+1}]$ and $f(v)\neq s$. Therefore $f([u_{i+1},u_i])=\{s\}$ implying that $f(U)=\{s\}$ as $[u_{i+1},u_i]\cap U= U$. □

Fig. 16 is an algorithm to solve Problem 2". Essentially, the algorithm first

Algorithm 2'

(1) Use Algorithm 2 to decide if there exists a composite function f_0 such that $|f_0(S)|=1$. If so, halt and report function.

(2) Construct a transition diagram for triples in S as was done for pairs in Algorithm 2.

(3) Let $\sigma=s_1,s_2,\ldots,s_n$ be an ordered enumeration for S and let $[u_1,u_2]$, $[u_2,u_3]$,...$[u_j,u_1]$ be the partitioning of σ by U. Find a path in the transition diagram above from a vertex (u_i,a,u_{i+1}) to a vertex (s,b,s) where $a\in\{[u_i,u_{i+1}]-U\}$ and $b\neq s$.

(4) The labels on this path form the required composition sequence. If no such path exists, report failure.

Fig. 16. A modified function composition algorithm

checks to see if there exists a composite function that maps all of S to a single element. If such exists, it reports the function. Else, any composite function that maps U to a singleton {s} must map at least one element a of S-U to some element other than s. Hence the algorithm searches for a composite function that satisfies conditions (a) and (b) of Lemma 2 on some interval $[u_i,u_{i+1}]$, reporting failure if none exists.

Time Analysis

Step (1) takes $O(kn^4)$ time as analysed earlier.

Step (2) takes $O(kn^3)$ time.

Step (3) can be performed in $O(kn^3)$ time using depth first search. The sequence length computed is $O(kn^3)$.

Total time is $O(kn^4)$ and the sequence length is $O(kn^3)$.

Remarks

Algorithm 2 finds the minimal cardinality set of orientations reachable from the set of all orientations. Algorithm 2' can be easily modified to to do so as well. This is bit more than what we wanted it to do and might prove useful to deduce modifications to the geometry of the object so as to permit unique orientation.

We assumed that all transitions were single valued. It would be interesting to study the problem with this restriction removed.

Using the notion of monotonicity, we can modify Algorithm 2 to produce a sequence of length $O(kn^2\log n)$ in $O(kn^3\log n)$ time. In particular, the algorithm halves the number of posible orientations at each iteration.

Earlier we stated without discussion that if a continuous spectrum of tilt directions were available, the problem would be simplified. We now discuss this issue. Consider the object shown in Fig. 17. Each side subtends two *stability* angles at the centroid, between the perpendicular through the centroid and the line joining the centroid to the vertices of the side. Let $L=\{l_1,l_2,\ldots,l_n\}$ be the left stability angles for the n sides of the object and let $R=\{r_1,r_2,\ldots r_n\}$ be the right stability angles as shown in Fig. 17. Suppose R (or L) had a distinct maximum as r_1 is for the object of Fig. 17. Then there exists a tilt direction for which there is exactly one stable orientation – the object lying on side 1. In particular, this is any angle between the two largest elements of R. Consequently, tilting in this direction will immediately orient the object uniquely making the problem rather simple. However, for objects for which neither R nor S have distinct maxima, we can formulate a new scheme which utilizes the stability angles to orient the object in a more direct way than the pan handler. The statement of the problem would read about the same though as here we have done little more than develop or 'unravel' the tray.

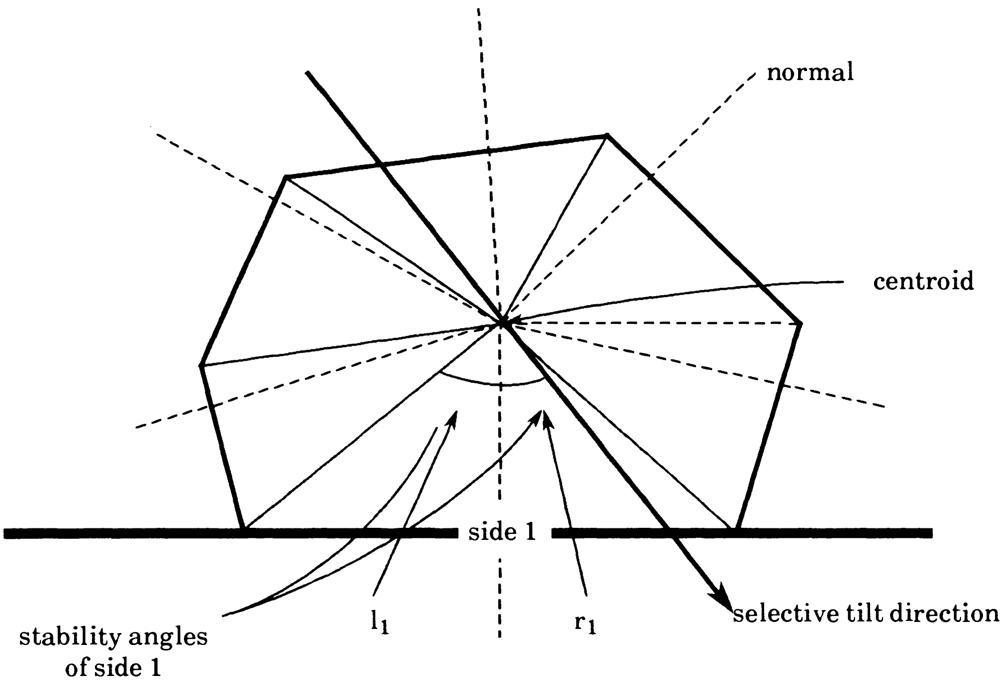

Fig. 17. Stability Angles.

Another question that is interesting is whether there exists a tray shape that is good for all or most objects. For any given tray, there exists a simple object that is not uniquely orientable by that tray using only median tilts, although there exists a tray that can do so. To see this, define the *incidence angles* of a tray to be the angles made by the tilt directions with the sides on which they are incident. See Fig. 18. Let α_1 be the smallest non zero incidence angle of the given tray. Pick an even integer N such that $\alpha = \pi/N$ lies between 0 and α_1 and construct an 'almost regular' polygon as shown in Fig. 19. In particular, pick $\varepsilon > 0$ such that $0 < \alpha - \varepsilon < \alpha < \alpha + \varepsilon < \alpha_1$ and construct a polygon whose vertices lie on a circle and whose sides subtend $2(\alpha - \varepsilon)$ and $2(\alpha + \varepsilon)$ at the centre alternately. See Fig. 19. Now the object has two different orientations, but for every median tilt direction of the given tray, both these orientations behave identically. Consequently, the given tray cannot reduce the set of possible orientations to either of the two. Yet, a tray with an incidence angle of α will uniquely orient the object in the orienatation in which it rests on the larger side. To see this, we only

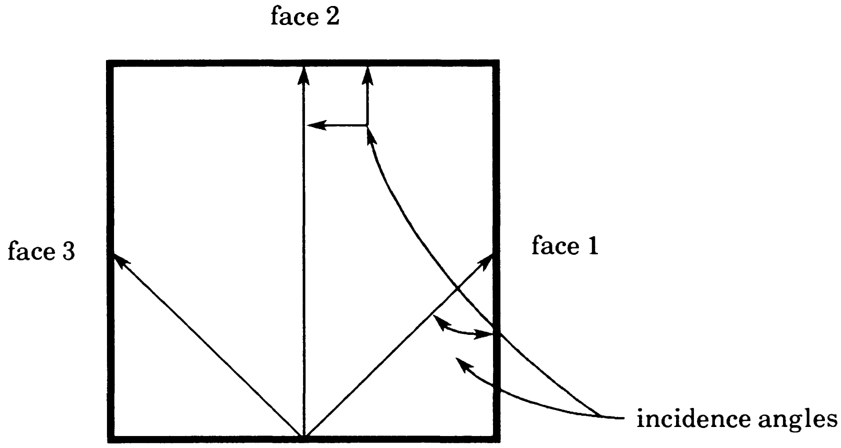

Fig. 18. Incidence angles

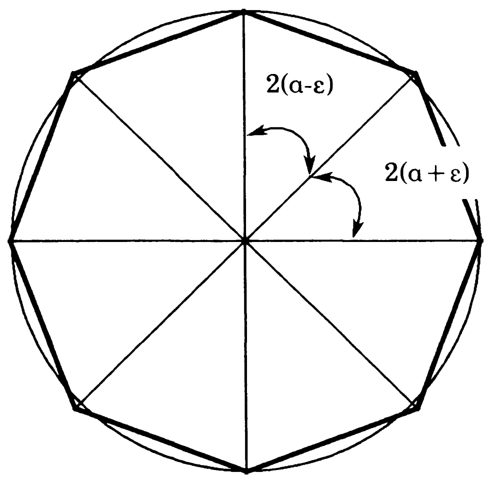

Fig. 19. An almost regular object

need notice that at an incidence angle of α, one of the orientations is unstable and hence all transitions must be to the other.

3. PARADIGMS FOR FILTERS

3.1 A First Attempt

We introduced the notion of a filter in Section 1.3. At first glance, the design of filters appears to lend itself to the approach that parts orienter design is the dual of motion planning. In fact, such an approach is explored in [Lozano-Perez, 1986]. I contend that such an approach is not very effective with support from the following example.

Example 1

Consider the planar rectangular object of Fig. 20a. Suppose we wish to orient it uniquely in one of its two orientations using planar operations. Consider a simple filter for the object where the object moves in a straight line along the x-axis and static obstacles are placed along the path so as to reject one of the two orientations of the object. We can depict the effect of the filter using the configuration space of the object - a two dimensional space (x,θ), x representing the position of the object along the x-axis and θ the orientation of the object. Since the object can only be in one of two orientations when it enters the filter, θ can be one of two values, say, $\theta = 0$ and $\theta = \pi/2$. See Fig. 20b. Now, if we were to view the

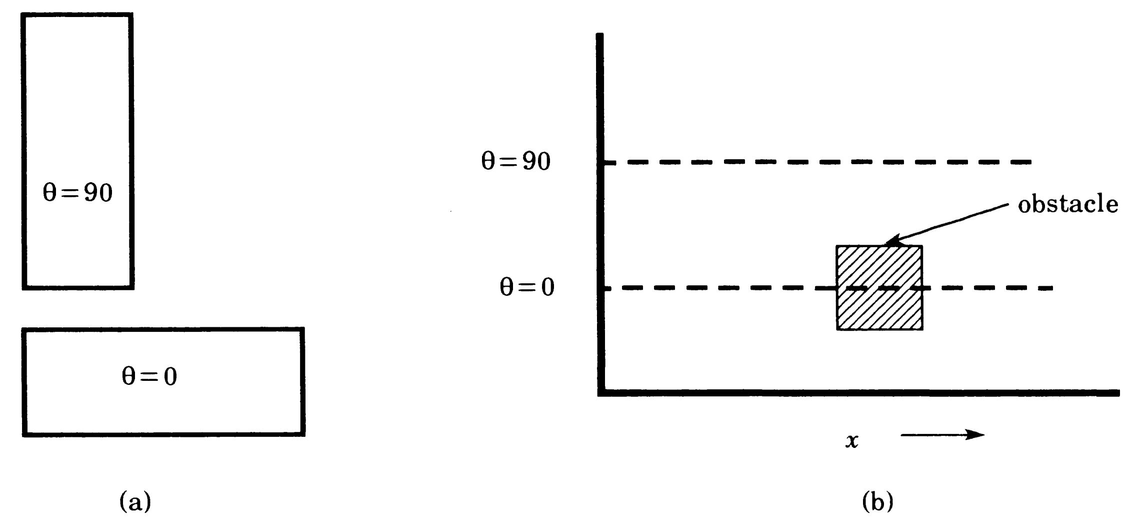

Fig. 20. (a) The rectangular object and its two orientations.
(b) The configuration space of the filter

design of a filter to be the inverse of motion planning, we need only pick an obstacle in this configuration space that intersects say the $\theta=0$ configuration line, but not the other. Mapping this configuration space obstacle to a physical obstacle completes the design of a filter that allows objects entering in the $\theta=90$ orientation to pass through but rejects the $\theta=0$ orientation. The difficulty here is that every configuration space obstacle does not correspond to a physical obstacle. In particular, the obstacle that we chose in Fig. 20b is not generated by any physical obstacle. This is to be expected as the configuration space of an object is typically a higher dimensional entity than the physical domain and the mapping from configuration space to physical space is singular.

Apart from such considerations, static design of the obstacles as discussed above will cause the filter to be blocked each time the object enters in an undesirable orientation. Designing obstacles to discard undesirable orientations cannot be done without knowledge of the dynamics of the situation. □

We are forced to conclude that the dynamic issues in parts orienting are so strong that simple mathematical formulations of the problem are infeasible. Consequently, we resort once again to considering specific paradigms.

3.2 The Vibratory Track

Here we consider the problem of orienting a convex, planar polygonal object on a sloped vibratory track with a lip as shown in Fig. 21. The object creeps along the track with one of its sides against the lip. The object may be reoriented by means of pins placed along the track and undesirable orientations may be discarded by means of cut-outs in the track. Assuming that the probability distribution of the orientation of the object entering the track is known *a priori*, we are to find a sequence of pin locations and cut-outs so that the object exits the track in some unique and predetermined orientation with maximal probability.

Problem Statement and Abstraction

In order to make a precise statement of the problem, we make the following assumptions:

(a) Some side of the object is always in contact with the lip of the track. This determines the orientation of the object.

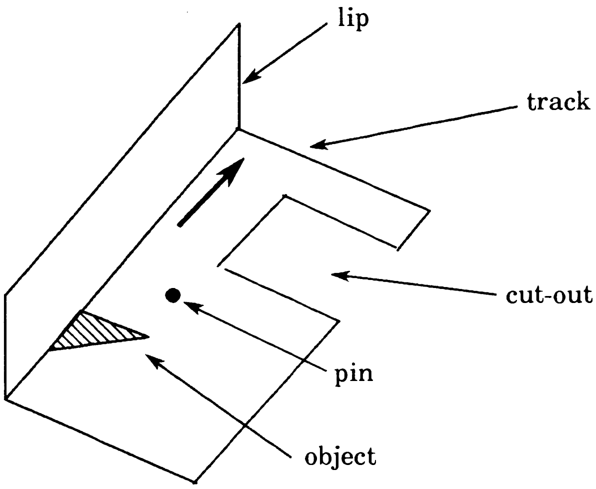

Fig. 21. The track.

(b) A cut-out runs parallel to the lip and is specified by the distance of the edge of the track from the lip at the cut-out. It has the following effect on an orientation: If the centroid of the object is outside the track, the object falls off the track and is discarded. Otherwise, the object is unaffected.
(c) A pin has the following effect on an orientation: Referring to Fig.22, if the

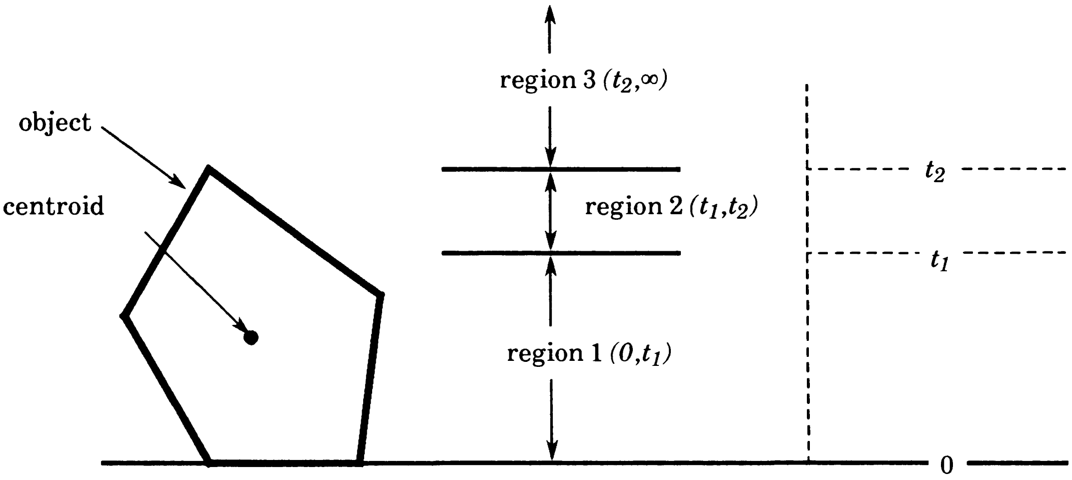

Fig. 22. The transition regions of an object

pin is located in region 1, the object gets stuck at the pin and cannot proceed.

This is to be avoided. If the pin is in region 2 (end points excluded), the object changes orientation to the next (clockwise or anti-clockwise order) stable orientation and proceeds past the pin. In region 3, the object proceeds past the pin without suffering a change in orientation. We refer to these three regions as the *transition regions* defined with respect to each orientation of the object and represent them by the distance intervals from the lip – $(0,t_1)$, (t_1,t_2) and (t_2,∞) as shown in Fig. 22. Fig. 22 is only indicative and not definitive.

(d) No pin on the track should be in region 1 of any of the possible orientations of the object at that point on the track.

(e) All operations are planar in that one of the two planar faces of the object is distinguished and is always in contact with the track.

We are now ready to state the problem:

Problem 3

Given a planar, convex, polygonal object, the transition regions for each orientation of the object, and the probability distribution of the orientation of the object as it enters the track, find a sequence of pin locations and cut-outs so that the object reaches the end of the track in some unique orientation with maximum probability.

Define the transition diagram of the object to be a directed, labeled graph with one vertex per orientation. An edge (u,v) with label $[t_1,t_2]$ is present in the graph if and only if $(0,t_1)$, (t_1,t_2) and (t_2,∞) are the transition regions for orientation u and orientation u changes to v when traversing a pin in the range (t_1,t_2). A vertex has label d if the centroid of the object is at a distance d from the base in that orientation. Observe that the transition diagram of any object consists of a single cycle. We can now state the problem as a graph pebbling game on the transition diagram.

Problem 3'

<u>Pebbling Game</u>

start: All vertices are pebbled. The weight of the pebble on a vertex represents the probability of the object entering the track in that orientation.

move: (a) pick a value $t>0$ such that if u *is a* pebbled vertex and the label on its outgoing edge is $[t_1,t_2]$, $t>t_1$. For each pebbled vertex u with outgoing edge (u,v) with label $[t_1,t_2]$, if $t<t_2$, move the pebbles on u to v.

(b) pick a value $d_0>0$ and discard the pebbles on every vertex with a label $d>d_0$.

Question: Find a sequence of moves that reduces the number of pebbled vertices to one, but maximizes the weight of the pebbles remaining on the graph at the end of the game over all sequences that result in a single pebbled vertex.

Fig. 23 gives an algorithm to solve the above problem. In the following we will assume that no edge has a trivial label ($[t_1,t_2]$ is trivial if $t_1=t_2$). This makes the algorithm technically simpler while allowing a straightforward extension to the general problem.

If the algorithm is successful, the moves in Phase 1 together with the moves in Phase 2 form the required sequence. Else no solution exists. □

Time Analysis

Phase 1 takes $O(n^3)$ as there are n^2 moves played and each move costs $O(n)$.

Phase 2 costs $O(n^4)$:

$O(n^2)$ to construct the pairwise diagram.

$O(n^3)$ for each iteration of the while loop with at most n iterations.

There are $O(n^3)$ moves in the solution sequence.

Correctness

Let t_{max} be the maximum label value in the transition diagram. Consider the set

$S=\{u|$ edge out of u has label $[t_1,t_{max}]$ for some $t_1\}$

and the related family of sets defined by

$S_u=\{v|$ $u\in S$, $v\notin S$ and there is a path from v to u in the transition diagram that does not include an element of $S\}$.

Finally define the equivalence relation $\equiv$ on S by $x\equiv y$ if $x, y\in S_u$ for some u.

Claim 2: If $a\equiv b$ then no sequence of moves can cause the pebbles that started on a and b to occupy the same vertex.

Proof: Follows from the observation that the transition diagram consists of a single cycle. □

Algorithm 3

Phase 1

Play the pebbling game for n^2 moves;

At each move

(a) pick t to be the least value that is legal.

(b) pick $d_0 = \infty$;

Phase 2

Construct a pairwise transition diagram for the object as follows. Each vertex is a pair of orientations (a,b). Edges are defined in the obvious way: if there is an edge from a to c labelled $[t_1,t_2]$ and b to d labelled $[q_1,q_2]$ respectively in the transition diagram then the edges to be included in the pairwise diagram are going to depend on the relationship between the label values. For example, suppose $q_1 \geq t_1$ and $t_2 \geq q_2$. Then the edges to be included are

(a,b) to (c,b) with label $[t_1,q_1]$

(a,b) to (c,d) with label $[q_1,q_2]$

(a,b) to (a,d) with label $[q_2,t_2]$.

If any of the above label intervals are trivial, the corresponding edge is omitted.

Colour a vertex (a,b) red if label(a) $<$ label(b) else colour it blue.

while there is more than one pebbled vertex on the board **do**

let $v_1,v_2,\ldots,$ be the pebbled vertices on the board in decreasing order of pebble weight.

Find a path by depth-first search from (v_1,v_2) to a red vertex (a,b). If no such exists halt and report failure.

With $d_0 = \infty$ play the pebbling game so that (v_1,v_2) traces this path. Then with $t = \infty$ and d_0 such that label(a) $< d_0 <$ label(b) play a move.

od

Fig. 23 The pebbling algorithm for the vibratory track.

Given this, we see that if at the end of some sequence of moves there is only one pebbled vertex, then all the pebbles on this vertex started on vertices in the same set S_v, for some vertex v. We now proceed by proving the following claims.

Claim 3: If there exists a sequence of moves σ that terminates with one pebbled vertex, then Algorithm 3 finds such a sequence.

Proof: Let σ terminate with a pebble starting on vertex x remaining on the board. Distinguish this pebble from all others. Now, by Claim 2, any pebble starting on say some vertex y such that $x \equiv y$ must be discarded by σ. At the time this pebble was discarded, say it occupied vertex b and the distinguished pebble occupied vertex a. Then, in our pairwise graph, (a,b) must be a red vertex.

At any stage in Phase 2 of the algorithm, let v_1, v_2, ..., be the pebbled vertices in decreasing order of pebble weight. Notice that Phase 1 ensures that $v_1 \equiv v_2 \equiv v_3$.... Furthermore, there exists a sequence of moves (with trivial d_0 values) that moves the pebbles on v_1 to x, as every edge in the transition diagram has a non-trivial label. Let this sequence move the pebbles on v_2 to some vertex y. Then $x \equiv y$ and the existence of σ ensures that there exists a path from (x,y) to a red vertex in the pairwise diagram. □

Claim 4: Algorithm 3 finds the sequence of moves that terminates with maximum pebble weight.

Proof: Let

$$w_{\max} = \max_{v \in S} \quad (w_v = \sum_{u \in S_v} \text{weight of pebble starting on } u)$$

That is, w_v is the weight of all the pebbles that start on vertices in S_v and w_{max} is the maximum of these. By Claim 2 w_{max} is the maximum weight that can be attained on a single vertex. If Algorithm 3 terminates successfully, it does so with w_{max}. Hence the claim is proved. □

Remarks

The paradigm of Section 3.2 and the subsequent analysis are admittedly simple. For one, three dimensional objects pose a greater challenge and it is unclear whether our anlysis of two dimensional objects is extendible to them. For another, our restrictions on the transitions of the object and the type of obstacles in the paradigm are rather strong.

4. Conclusion

We showed how to abstract seemingly ill-defined physical problems to the computational domain in the setting of the design of parts orienters. While the paradigms considered were admittedly simple, it leads one to believe that abstract analysis of such problems will reveal their rich structure, at the same time providing insight into the physical domain. In short, I hope this paper does more than analyze parts feeders.

5. References

Boothroyd, G., Poli, C, and Murch L.E., 1982, Automatic Assembly, Mercel-Dekker.

Grossman, D.D. and Blasgen, M.W., 1975, Orienting Mechanical Parts by Computer Controlled Manipulator, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC 5, No. 5, pp561-565.

Kozen, D., 1977, Oct. 31-Nov. 2, Lower Bounds for Natural Proof Systems, 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island.

Lozano-Perez, T., 1986, Private communication.

Lozano-Perez, T., and Wesley. M.A., 1979, An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles, Communications of the ACM, Vol. 2, No. 10, pp560-570.

Mason, M.T., and Erdmann, M., 1986, April, An Exploration of Sensorless Manipulation, 3rd IEEE International Conference on Robotics and Automation, San Francisco, California.

Philip, T.K., 1980, Lecture Notes on Automated Manufacturing, Dept. of Mechanical Engineering, Indian Institute of Technology, Madras.

Reif, J.H., 1979, Oct. 29-31, Complexity of the Mover's Problem and Generalizations, 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico.

Schwartz, J.T., and Sharir., M., 1982, On the 'Piano Mover's' Problem II, Courant Institute of Mathematical Sciences, New York, Report No. 41.

Udupa, S., 1977, Collision Detection and Avoidance in Computer Controlled Manipulators, Ph.D. Thesis, Cal.Tech, Pasadena, California, 1977.