A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles

Jean-Daniel Boissonnat, Bernard Faverjon, Francis Avnaim

INRIA

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 890

A PRACTICAL EXACT MOTION PLANNING ALGORITHM FOR POLYGONAL OBJECTS AMIDST POLYGONAL OBSTACLES

Francis AVNAIM
Jean Daniel BOISSONNAT
Bernard FAVERJON

AOUT 1988

RR_0890

A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles

Un algorithme exact pour la planification de trajectoires d'un objet polygonal dans un environnement polygonal

Francis Avnaim, Jean Daniel Boissonnat and Bernard Faverjon

INRIA
Centre de Sophia-Antipolis
Route des lucioles
06565 Valbonne

Abstract:

Let I be a 2-dimensional polygonal rigid object (with m edges) moving amidst polygonal obstacles E (with n edges) and let P_{init} and P_{end} be two free placements of I, where the interior of I does not intersect E. We investigate here the problem of finding a continuous motion of I from P_{init} to P_{end}, such that during this motion the interior of I does not intersect E, or to establish that no such motion exists. This problem is an instance of the well known "Piano Movers' Problem". We have shown in [2] that it is possible to compute an exact description of free space in time $O(m^3n^3\log(mn))$. We show in this paper that, using this description, a motion can be found in time $O(m^3n^3)$. The actual complexity of our algorithm in many practical situations is much smaller. In particular, for the so called situation of local bounded complexity often encontered in robotics, the complexity of computing free space is $O(n\log n)$ and the complexity of planning a motion is $O(n)$. The method has been implemented and experimental results are discussed.

Résumé:

Soit I un objet polygonal rigide ayant m côtés évoluant dans un environnement polygonal E comportant n côtés. Soit P_{depart} et P_{but} deux placements libres pour I, c-a-d tels que l'intérieur de I n'intersecte pas les obstacles E. Nous nous intéressons ici au problème suivant: calculer, s'il en existe un, un mouvement continu entre P_{depart} et P_{but} de telle manière que durant ce mouvement, l'intérieur de I n'intersecte pas les obstacles. Ce problème est une version du problème général connu dans la littérature sous le nom du problème du "déménageur de piano". Nous avons montrés dans [2] qu'il est possible de calculer une description exacte de l'espace libre en temps $O(m^3n^3\log(mn))$. Nous montrons dans ce rapport que, utilisant cette description, il est possible de calculer un tel mouvement en temps $O(m^3n^3)$. La complexité de l'algorithme est meilleure dans un grand nombre de situations pratiques. En particulier, dans le cas du robot mobile la complexité du calcul de l'espace libre est de $O(n\log n)$ et celle du calcul d'un mouvement est de $O(n)$. L'algorithme a été implanté et les résultats expérimentaux sont discutés.

1 Introduction

We investigate here the problem of planning the motion of a 2-dimensional polygonal rigid simply connected object I (with m edges) which is free to move by translation and rotation amidst polygonal obstacles E (with n edges). More specifically, given two placements P_{init} and P_{end}, we want either to find a continuous motion connecting P_{init} and P_{end} during which the interior of I avoids collision with E, or else establish that no such motion exists.

This problem has been attacked from both practical and theoretical points of view in the literature. The only solutions that have been implemented use heuristic approaches: in [8] an approximation of the set of free placements (the so called free space) is computed and then a path is searched. Due to the approximation done, we are not guaranted to always find a path if one exists. In [3] the motion of I is restricted to be a sequence of pure translational and pure rotational movements [3]. Here again, we are not guaranted to always find a path if one exists. The first exact solution to the problem is due to Schwartz and Sharir [11]. The complexity of their algorithm is $O(n^5)$ in the case that I is a line segment (a *ladder*). This algorithm is rather involved and, according to the authors themselves, several technical delicate issues are ignored. Recently, Kedem and Sharir [7] improved on this result in the case that I is convex. The complexity of their algorithm is $O(mn\lambda(mn)\log(mn))$, where $\lambda(q)$ is an almost linear function of q. The improvement is obtained by only computing a judicious subset of the set of free placements, namely a set of edges on its boundary. This is sufficient to find a path if one exists but the computed path may be very unsatisfactory in practice since, during the motion, I keeps at least two points in contact with E.

In this paper, we propose a rather simple method to handle the general case where I may be non convex. Our algorithm makes use of the exact description of the boundary of free-space obtained in [2]. This description consists of a set of "faces" that are portions of ruled surfaces. There are $O(m^3n^3)$ faces in the worst case. The faces and their adjacency relationships can be computed in time $O(m^3n^3\log(mn))$ in

the worst case and much faster in many practical situations [2]. For example, when the number of vertices of I is small (and thus can be considered as a constant) and when, in addition, the edges of E are not concentrated near each other compared with the diameter of I –a situation refered to as of local bounded complexity [12]– the boundary of free space is computed in $O(n \log n)$ time. In this paper, we show that this description can be used to solve the motion planning problem. The time complexity is in the worst case proportional to the number of faces which is $O(m^3n^3)$. For situations of local bounded complexity, the time complexity is only $O(n)$. An important aspect of our method is that the motions produced are searched in a 2-dimensional variety thus allowing to locally modify and optimize them. The method has been implemented and experimental results are discussed.

2 Representation of the boundary of free-space

Let I be a 2-dimensional polygonal rigid simply connected object with m edges moving amidst a set E of polygonal obstacles with a total number of n edges. We assume that the complement of E is a bounded polygonal region (this is always possible by enclosing obstacles in a sufficiently large rectangle) (see Fig. 1). *Free-space* FP is defined to be the closure of the subset of $[0, 2\pi[\times R^2$ consisting of all placements $(\theta, \vec{u})$ satisfying $R_\theta \circ T_{\vec{u}}(I) \cap E = \emptyset$ where R_θ denotes rotation with center at the origin and angle θ and $T_{\vec{u}}$ denotes translation by vector $\vec{u}$. A placement of FP is called a *free placement*. An algorithm has been described in [2] which computes a complete description of the boundary BFP of FP. This description consists of a disjoint union of "faces" and of their adjacency relationships. Each face is a portion of a ruled surface generated by a line segment $P(\theta)Q(\theta)$ when θ ranges in a subinterval $[\theta_{min}, \theta_{max}]$ of $[0, 2\pi[$ (see Fig. 2). Points $P(\theta)$ and $Q(\theta)$ are of type $(\theta, f(\theta), g(\theta))$ where $f(\theta)$ and $g(\theta)$ are analytic functions of θ. Each face is a set of placements involving a given contact between I and E (i.e., a given vertex of I in contact with a given edge of E or a given vertex of E in contact with a given edge of I). There are $O(m^3n^3)$ faces in the worst case.

Each face is bounded by at most four edges; two edges are (portions of) the curves $P(\theta)$ and $Q(\theta)$ for $\theta \in [\theta_{min}, \theta_{max}]$. The two others are the line segments $P(\theta_{min})Q(\theta_{min})$ and $P(\theta_{max})Q(\theta_{max})$, which may be reduced to points. Two faces are adjacent if they share an edge or a portion of an edge. In the sequel, we will represent BFP by a graph called the *boundary graph* of FP and denoted by $\mathcal{G}$. The nodes of $\mathcal{G}$ are the faces of BFP and its edges join adjacent faces. It is shown in [2] that the size of this graph is $O(m^3n^3)$ and that it can be computed in time $O(m^3n^3\log(mn))$ in the worst case.

In many practical situations, the number of faces composing the boundary of FP is much smaller than in the worst case. For example, when the number of vertices of I is small (and thus can be considered as a constant) and when, in addition, the edges of E are not concentrated near each other compared with the diameter of I –a typical situation in robotics refered to as a situation of local bounded complexity– the boundary of FP has $O(n)$ faces and can be computed in time $O(n\log n)$.

The intersection FP_{θ_0} between FP and the plane $\theta = \theta_0$ (for any fixed orientation θ_0) is a polygonal region which is the set of free placements when I can only move by translation with fixed orientation θ_0. It is shown in [1] that such a polygonal region has $O(m^2n^2)$ edges in the worst case and can be computed in $O(m^2n^2\log(mn))$ time. Moreover, any of the connected components of FP_{θ_0} has $O(mn\alpha(mn))$ edges where $\alpha(mn)$ is the functionnal inverse of Ackermann's function, and thus is extremely slowly growing [9]. It is plain to compute the boundary of FP_{θ_0} from graph $\mathcal{G}$ in time $O(m^3n^3)$ (a better result will be given in Section 4.3).

3 Computing a free path

Let $P_{init} = (\theta_{init}, X_{init}, Y_{init})$ and $P_{end} = (\theta_{end}, X_{end}, Y_{end})$ be two free placements of I. We want to find a continuous obstacles avoiding motion of I between P_{init} and P_{end}. This is equivalent to searching a curve inside FP joining P_{init} and P_{end}. Such a curve is called a *free path* from P_{init} to P_{end}. We successively study the three following instances of the problem (with increasing difficulty):

1. P_{init} and P_{end} belong to the same face of BFP.

2. P_{init} and P_{end} belong to the same connected component of BFP.

3. P_{init} and P_{end} are in general position.

Case 1: Let f be a face of BFP and A and B two points of f. As f is a ruled surface swept by a line segment $P(\theta)Q(\theta)$, A (resp., B) is completely defined by an orientation θ_A (resp., θ_B) and a real α_A (resp., α_B) such that $\overrightarrow{P(\theta_A)A} = \alpha_A\overrightarrow{P(\theta_A)Q(\theta_A)}$ (resp., $\overrightarrow{P(\theta_B)B} = \alpha_B\overrightarrow{P(\theta_B)Q(\theta_B)}$). It is plain to observe that the curve Γ defined by

$$\Gamma = \{M(\theta), \overrightarrow{P(\theta)M(\theta)} = (\alpha_A + (\alpha_B - \alpha_A)\frac{\theta - \theta_A}{\theta_B - \theta_A})\overrightarrow{P(\theta)Q(\theta)}, \theta \in [\theta_A, \theta_B]\}$$

passes through A and B and lies entirely inside face f (see Fig. 3). Thus Γ is a free path between A and B. Notice that if face f corresponds to contact C, this contact will be maintained all along Γ.

Case 2: When P_{init} and P_{end} belong to the same connected component of BFP (but not to the same face), we can search in the boundary graph of FP a sequence S of faces $f_1, \cdots, f_k$ such that f_1 contains P_{init}, f_k contains P_{end} and f_i is adjacent to f_{i+1} $(i = 1, \cdots, k-1)$. Let us consider two faces adjacent in the sequence S, say f_i and f_{i+1}. We associate to f_i and f_{i+1} a point P_{ii+1} belonging to the two faces. If f_i and f_{i+1} are adjacent by a segment, P_{ii+1} is simply the middle of the segment. If they are adjacent by a portion of curve ranging from θ_1 to θ_2 , P_{ii+1} is the point on this curve corresponding to $\frac{\theta_1+\theta_2}{2}$ (see Fig. 4). Thus the sequence S yields a sequence of points $P_{init}, P_{12}, \cdots, P_{k-1k}, P_{end}$ such that P_{init} and P_{12} belong to f_1, P_{i-1i}, P_{ii+1} belong to f_i (for $i = 2, \cdots, k-1$), P_k and P_{end} belong to f_k. As any two consecutive points in that sequence belong to the same face it is possible to compute a curve between them lying on that face (Case 1). Thus the concatenation of theses curves is a free path between P_{init} and P_{end}. This path is entirely contained in BFP

and thus corresponds to a motion where I remains in contact with E. Clearly, the dominant step in the above procedure is the search of a sequence of faces of BFP. This search can be done in time proportional to the size of the graph, which is in turn proportional to number of faces of BFP (see Section 2).

Case 3: In that case, P_{init} and P_{end} belong to the interior of FP. Due to the fact that the complement of E is bounded (see Section 2), FP is also bounded. Let us denote by FP^{init} (resp., FP^{end}) the connected component of FP containing P_{init} (resp., P_{end}). The boundary of FP^{init} (resp., FP^{end}) has several connected components, one of them enclosing the others. We call it the *external boundary* of FP^{init} (resp., FP^{end}) and denote it by FP^{init}_{ext} (resp., FP^{end}_{ext}). There exists a free path from P_{init} to P_{end} iff P_{init} and P_{end} belong to the same connected component of FP, i.e., iff $FP^{init} = FP^{end}$. As in Section 2, let FP_θ be the set of free placements for a fixed value θ of the orientation of I. Let us call U^{init} (resp., U^{end}) the union of the boundary of FP^{init} and of the polygonal region $FP_{\theta_{init}}$ (resp., the union of the boundary of FP^{end} and the polygonal region $FP_{\theta_{end}}$). As FP^{init} (resp., FP^{end}) is bounded and connected, there exists a continuous path in U^{init} (resp., U^{end}) which joins point P_{init} (resp., P_{end}) and the external boundary of FP^{init} (resp., FP^{end}). Indeed, let D be any half line lying in $FP_{\theta_{init}}$ with P_{init} as its endpoint. Let $S = \{S_1 = P_{init}, S_2, \cdots, S_{k-1}, S_k\}$ be the sequence of the intersection points between D and the boundary of FP^{init}, sorted along D. Note that S_k belongs to FP^{init}_{ext}. The interior of any segment S_iS_{i+1} is either inside FP^{init} or outside FP^{init}. Let S_iS_{i+1} be a segment such that its interior is outside FP^{init}. As FP^{init} is connected, there exists a path α_{ii+1} contained in the boundary of FP^{init} joining S_i and S_{i+1}. Thus the concatenation C_{init} of $S_1S_2, \alpha_{23}, S_3S_4, \cdots, S_{k-3k-2}, \alpha_{k-2k-1}, S_{k-1}S_k$ is a continuous path joining P_{init} and S_k. Similar arguments show that there exists a continuous path C_{end} in U^{end} and a point $S'_{k'}$ belonging to FP^{end}_{ext}such that C_{end} joins P_{end} and $S'_{k'}$. If there exists a path in FP joining P_{init} and P_{end} then $FP^{init} = FP^{end}$, thus FP^{init} and FP^{end} have the same external boundary. Moreover, as this external

boundary is connected, there exists a path C_{ext} contained in it that joins S_k and $S'_{k'}$. The concatenation of C_{init}, C_{ext} and C_{end} is a continuous path from P_{init} to P_{end} contained in the union of $BFP, FP_{\theta_{init}}$ and $FP_{\theta_{end}}$. Figure 5 illustrates such a construction (for clarity, D and D' have been taken to be coplanar). In conclusion, if there exists a path in FP joining P_{init} and P_{end}, then there exists a path in the union of the boundary of $FP, FP_{\theta_{init}}$ and $FP_{\theta_{end}}$ joining P_{init} and P_{end}. Reciprocally, any path in the union of the boundary of $FP, FP_{\theta_{init}}$ and $FP_{\theta_{end}}$ joining P_{init} and P_{end} is clearly a path in FP joining P_{init} and P_{end}.

We can deduce from the above discussion a method to compute a free path joining P_{init} and P_{end}. First we triangulate $FP_{\theta_{init}}$ and $FP_{\theta_{end}}$. Let $T_{\theta_{init}}$ and $T_{\theta_{end}}$ be the adjacency graphs of the two triangulations. Each edge of the boundary of one of the two triangulations belongs to a face of BFP. Let t be a triangle of a triangulation having an edge e belonging to a face f of BFP. We create an adjacency relation between f and t. Doing so for all possible triangles, we merge the graphs $\mathcal{G}$, $T_{\theta_{init}}$ and $T_{\theta_{end}}$ and achieve a new graph $\mathcal{G}^*$. As any triangle is a portion of a ruled surface (a plane) which accepts exactly the same description as a face of $\mathcal{G}$, the resulting graph $\mathcal{G}^*$ is a graph of faces such that P_{init} and P_{end} belong to faces of it. Moreover the existence of a free path from P_{init} to P_{end} is equivalent to the existence of a path in $\mathcal{G}^*$ from the triangular face containing P_{init} to the one containing P_{end}. Thus searching a free path when P_{init} and P_{end} are in general position reduces to Case 2 using graph $\mathcal{G}^*$.

We briefly describe the computation of a motion:

1. Deduce from graph $\mathcal{G}$ the polygonal region $FP_{\theta_{init}}$. Compute a triangulation of $FP_{\theta_{init}}$ yielding a graph T_{init} of triangular faces. Merge T_{init} and $\mathcal{G}$ by making an adjacency relation between a face t of T_{init} and a face f of $\mathcal{G}$ iff an edge of t belongs to f.

   The same is done for P_{end} yielding a final graph $\mathcal{G}^*$.

2. Search the two triangular faces f_{init} and f_{end} containing respectively the points P_{init} and P_{end}.

3. Search in $\mathcal{G}^*$ a sequence S of faces $f_1, \cdots, f_k$ such that $f_1 = f_{init}$, $f_k = f_{end}$ and f_i is adjacent to f_{i+1} $(i = 1, \cdots, k-1)$. Compute the corresponding sequence of points $P_{init}, P_{12}, \cdots, P_{k-1k}, P_{end}$. If this search is unsuccessful, return "no path".

4. Compute the k curves $\Gamma_1, \cdots, \Gamma_k$ such that Γ_1 is a curve inside f_1 joining the points P_{init} and P_{12}, Γ_i is a curve inside f_i joining the points P_{i-1i} and P_{ii+1} $(i = 2, \cdots, k-1)$ and Γ_k is a curve inside f_k joining the points P_{k-1k} and P_{end}.

5. Return the path obtained by concatening $\Gamma_1, \cdots, \Gamma_i, \cdots, \Gamma_k$.

Complexity analysis Let K_{init} (resp., K_{end}) be the number of edges composing the boundary of $FP_{\theta_{init}}$ (resp., $FP_{\theta_{end}}$) and F be the number of faces of BFP. Let K be the sum $K_{init} + K_{end}$. The complexity analysis is done in function of K and F.

1. $FP_{\theta_{init}}$ is computed in time $O(F)$ using graph $\mathcal{G}$. As $FP_{\theta_{init}}$ has at most $O(K_{init})$ edges, T_{init} can be computed in $O(K_{init} \log K_{init})$ time. Similarly, T_{end} can be computed in $O(K_{end} \log K_{end})$ time. Merging the graphs T_{init} and $\mathcal{G}$ takes $O(K_{init})$ time and merging the graphs T_{end} and $\mathcal{G}$ takes $O(K_{end})$ time. Therefore, the final graph $\mathcal{G}^*$ is computed in time $O(K_{init} \log K_{init} + K_{end} \log K_{end} + F) = O(K \log K + F)$.

2. The localisation of P_{init} and P_{end} in their respective triangulation takes $O(K)$ time since they are $O(K)$ triangles in each triangulation. The final graph $\mathcal{G}^*$ has $O(F)$ edges, thus a searching a path in $\mathcal{G}^*$ takes $O(F)$ time.

3. The computation of curves Γ_i takes $O(F)$ time.

We sum up the above results in the following proposition :

Proposition 1 *A free path between P_{init} and P_{end} can be computed in time $O(K \log K + F)$.*

In the worst case $K = O(m^2n^2)$ and $F = O(m^3n^3)$ thus the worst case complexity of the computation of a free path between P_{init} and P_{end} is $O(m^3n^3)$. For situations of local bounded complexity $K = F = O(n)$ thus the complexity of our motion planning algorithm is $O(n \log n)$.

4 Final remarks

4.1 Computing pseudo-optimal motions

Computing a free path from P_{init} to P_{end} has been reduced to a simple search in a graph. If the edges of these graphs are valuated by some real numbers, we can find a shortest path according to these values. The value associated to an edge joining two nodes can be, for example, the euclidean length of the corresponding curve. Depending on the choice of the unit respective lengths on the axis x, y and on axis θ, the resulting path will minimize preferably the translational movements or the rotational movements of I. This technique does not yield theoretic optimal paths but these paths appear to be reasonable in practice.

Searching such a pseudo optimal motion takes $O(M + N \log(N))$ where M is the number of edges and N is the number of nodes in the graph [6]. As $M = N = F$, a pseudo optimal motion can be found in time $O(F \log F + K \log K) = O(F \log F)$ which is $O(m^3n^3 \log(mn))$ in the worst case.

Furthermore, an important aspect of our method is that the produced motions can be locally modified and optimizied. Indeed, we have at our disposal a 2-dimensional variety (represented by graph $\mathcal{G}^*$). Although we have only used a finite set of curves in this variety (a 1-dimensional subvariety) to search a motion. Once a motion has been computed, we can locally improve this motion by relaxing the positions of the points $Pii + 1$ on their respective edges (see for analogous point of view [4].

4.2 Motion along a reference trajectory

In some applications, a point of reference on I is required to move along a given polygonal line while I may rotate around this point. Let k be the number of segments composing the polygonal line. In that case, we need only to compute the faces corresponding to the vertex-edge contacts involving the point of reference (which plays the same role as a vertex of I) and the k segments of the polygonal line. Refering to [2], it is easy to see that the computation of these faces takes $O(km^2n^2 \log mn)$ time and thus searching a free motion takes $O(km^2n^2)$ time.

4.3 Repeated queries for translationnal motion planning

Our description of FP can also be used to find repeated translationnal motions. This is done by constructing an appropriate data structure which allows efficient queries. To each face f of BFP corresponds an interval $[\theta_{min}, \theta_{max}]$ denoted by I_f. We denote by $\mathcal{I}$ the set of all the intervals corresponding to the faces of BFP.

We want to compute from boundary graph $\mathcal{G}$ of FP the boundary of the polygonal region FP_{θ_0}. Let K be the number of edges composing FP_{θ_0}. Any edge e belonging to the boundary of FP_{θ_0} belongs to exactly one face f of BFP. Moreover, the interval I_f contains orientation θ_0. Reciprocally, let f be any face in BFP generated by the segment $P(\theta)Q(\theta)$ when θ ranges in I_f, and assume that θ_0 belongs to I_f. It is clear that $P(\theta_0)Q(\theta_0)$ is an edge belonging to the boundary of FP_{θ_0}. Thus the set of edges composing the boundary of FP_{θ_0} is trivially deduced from the set L_{θ_0} of faces defined by:

$$L_{\theta_0} = \{f \in BFP, \theta_0 \in I_f\}$$

Note that L_{θ_0} has exactly K elements. We store the intervals of L in a segment tree [10]. This takes $O(F \log F)$ time as there are F intervals in L (note that this does not increase the time complexity of the computation of graph $\mathcal{G}$). Then using this segment tree, we compute in time $O(K + \log F)$ the set L_{θ_0} and thus the set of edges composing the boundary of FP_{θ_0}. To get a complete description of the

boundary of FP_{θ_0}, it remains to compute the adjacency relationships between these edges. This is done in time $O(K \log K)$ as follows. Each endpoint of an edge of the boundary of FP_{θ_0} is labelled by a double-contact (see [2]). Thus, after sorting these edges with respect to their endpoint 's labels –which takes $O(K \log K)$ time– we can easily compute in time $O(K)$ the adjacency relationships between the edges of the boundary of FP_{θ_0}. In conclusion, the boundary of FP_{θ_0} is computed in time $O(K \log K + \log F)$. Searching a translationnal motion between two points of FP_{θ_0} can then be done within the same time bound.

4.4 Implementation

The search of a motion has been implemented in C on a Sun workstation. Examples of free spaces and motions are shown in Figure 6a, 6b. The table below sums up the experimental results:

Exp. results	I	E	Faces	Computing FP (s)	Computing a free path (s)
Fig. 6a	10	12	613	270	0.5
Fig. 6b	6	12	285	92	0.3

The program consists of approximatively 15000 lines of C, including the computation of FP and the computation of a motion. First experiments have shown that computing FP is the most costly part of our method. It can be considered as a pre-processing.

Compared to the traditional approaches discretizing FP, our method has the important advantage of being exact. This means that very "difficult" paths (see Fig. 6a, 6b for examples) can be found. Moreover, the program is rather simple and the computing times and the complexity of the description of FP (number of faces) compare favourably with others approaches. For example, to solve the problem shown in Figure 6b, we have stored a graph with 270 nodes while Faverjon's method [5] requiers to store a graph of 10000 cells. It seems impossible to solve the problem

shown in Figure 6a with approximate or heuristique methods.

5 Conclusion

We have shown that, using a complete description of the boundary of free space FP(i.e., the set of all free placements for a polygonal object I (with m edges) which is free to translate and to rotate but not to intersect another polygonal object E), it is possible to compute a free motion for I between two free placements, if such a motion exists. We compute first and once for all the boundary of FP. This preprocessing takes $O(m^3n^3\log(mn))$ [2]. Then each free motion is computed in $O(m^3n^3)$ time. As we get a complete exact description of the boundary of free space, it is possible to compute many different motions between two free placements, according to diferent criteria. Last but not least, the algorithm has been implemented and first experimental results are very hopeful. The immediate application of our algorithm is the motion planning for a planar mobile robot moving amidst polygonal obstacles.

References

[1] AVNAIM F., BOISSONNAT J.D., Simultaneous containment of several polygons, 3rd ACM Symp. on Computational Geometry, Waterloo (June 1987).

[2] AVNAIM F., BOISSONNAT J.D., Polygon placement under translation and rotation, LNCS N. 294 Springer Verlag pp.322-333 (1988). To appear also in RAIRO Informatique théorique et applications 1988.

[3] BROOKS R.A., Solving the find-path problem by good representation of free space, IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-13 pp.190-197, (March-April 1983).

[4] CHANDERJIT BAJAJ, MOH T.T., Generalized unfoldings for shortest paths. The international journal of Robotics Research vol. 7 number 1 ISSN 0278-3649 MIT press (Feb. 1988).

[5] FAVERJON B., Obstacle avoidance using an octree. In Proceedings of IEEE Int. Conference on Robotics and Automation (March 1988).

[6] FREDMAN M., TARJAN R.E., Fibonacci heaps and their uses uin improved network optimization problems. In Proc. 25th IEEE FOCS, pp.338-346, (1984).

[7] KEDEM K., SHARIR M., An efficient motion planning algorithm for a convex polygonal object in 2-dimensional polygonal space, Tech. Rept. No 253, Comp. Sci. Dept., Courant Institute, (Oct. 1986).

[8] LOZANO-PEREZ T., BROOKS R.A., A subdivision algorithm in config uration space for findpath with rotation, IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-15 No 2, pp.224-233 (1985).

[9] POLLACK R., SHARIR M., SIFRONY S., Separating two simple polygons by a sequence of translations, Discrete and Computational Geometry 3:pp.123-136 (1988).

[10] PREPARATA F.P., SHAMOS M.I., Computational geometry: an introduction, Springer Verlag, (1985).

[11] SCHWARTZ J.T., SHARIR M., On the piano movers' problem: I. The special case of rigid polygonal body moving amidst polygonal barriers, Comm. Pure Appl. Math., Vol. XXXVI, pp.345-398 (1983).

[12] SIFRONY S., SHARIR M., A New Efficient Motion Planning Algorithm for a Rod in Two-Dimensional Polygonal Space, Algorithmica 2, pp. 367-402 (1987).

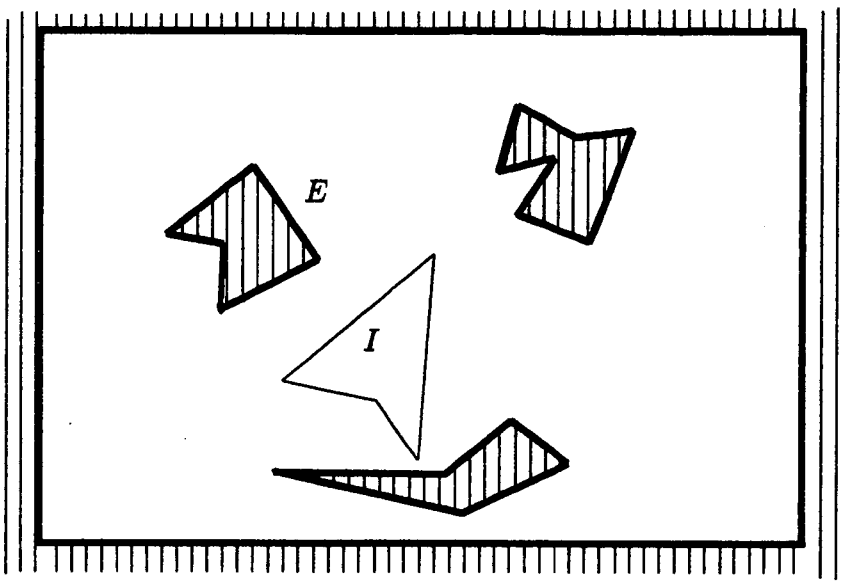


Figure 1: Polygon I and obstacles E

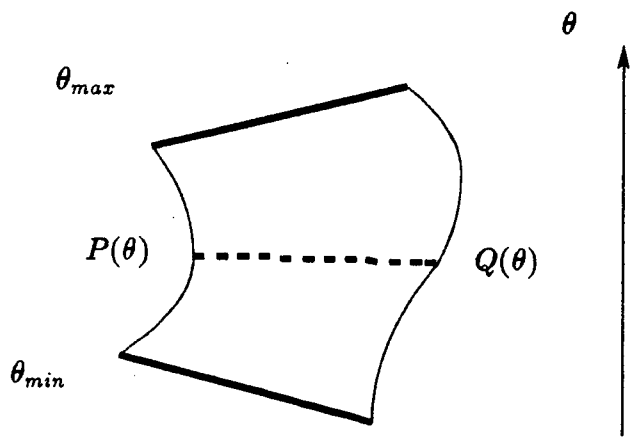


Figure 2: A face of BFP

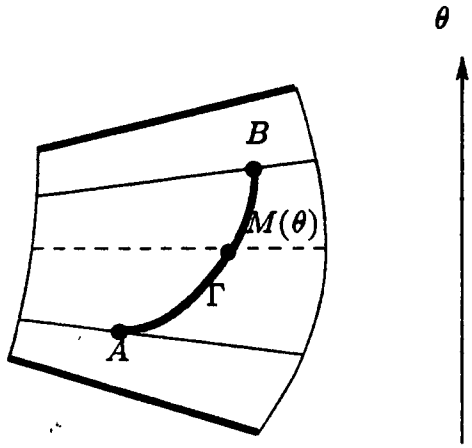


Figure 3: Curve Γ joining two points on a face of BFP

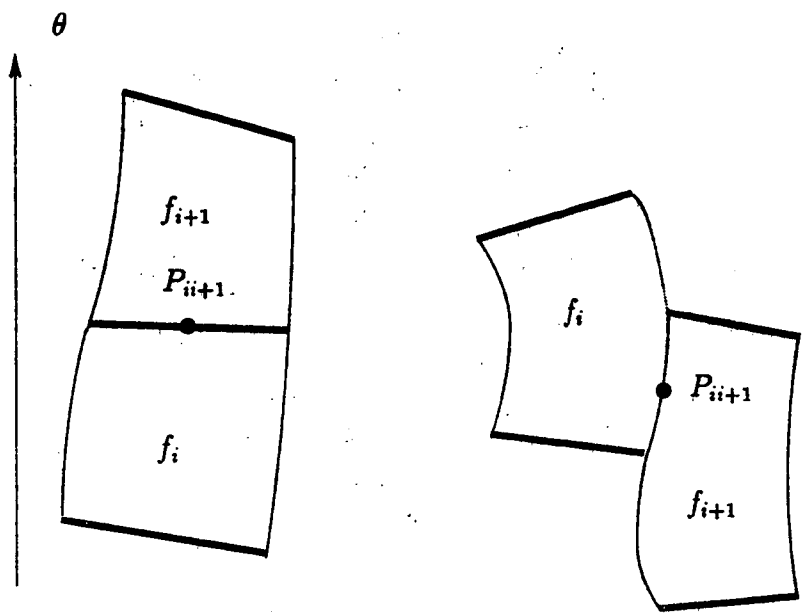


Figure 4: Common point of two adjacent faces

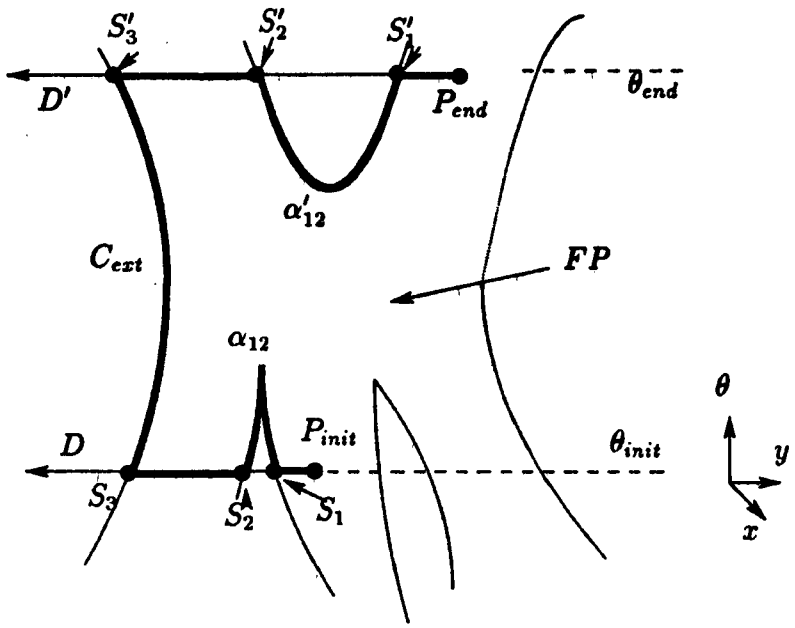
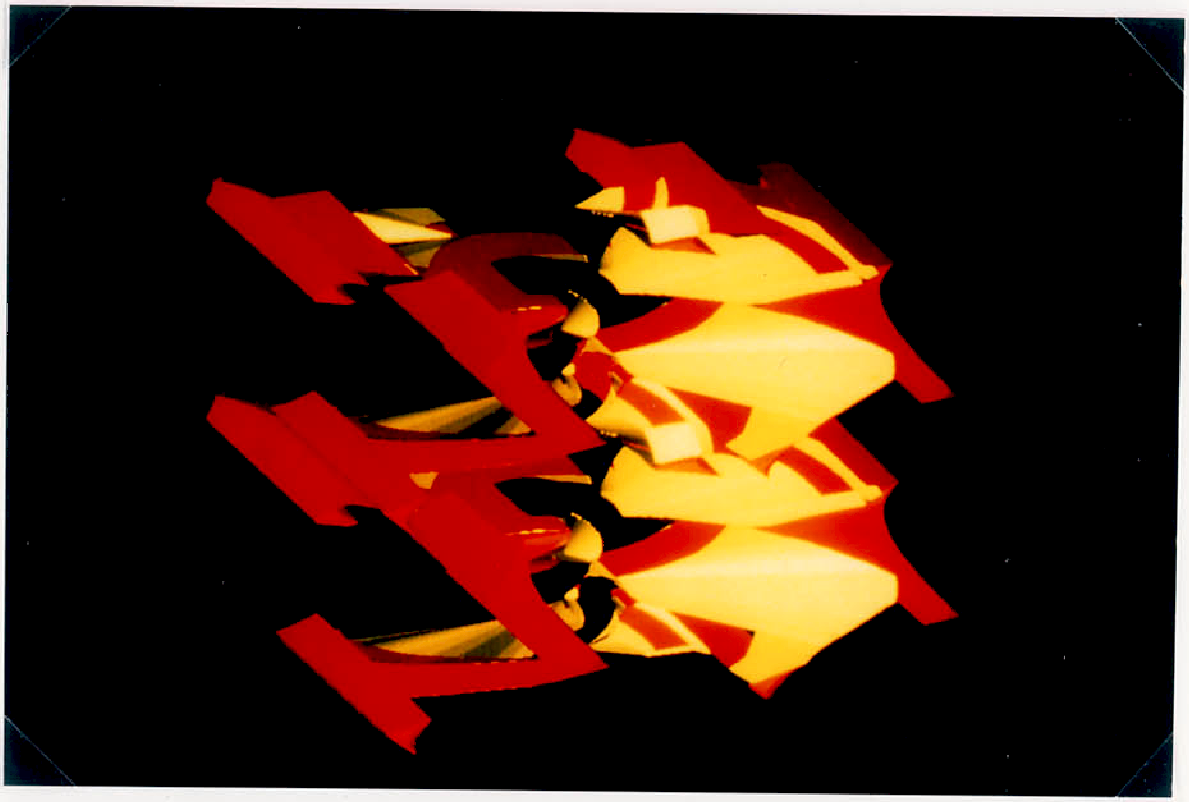


Figure 5: A path from P_{init} to P_{end}



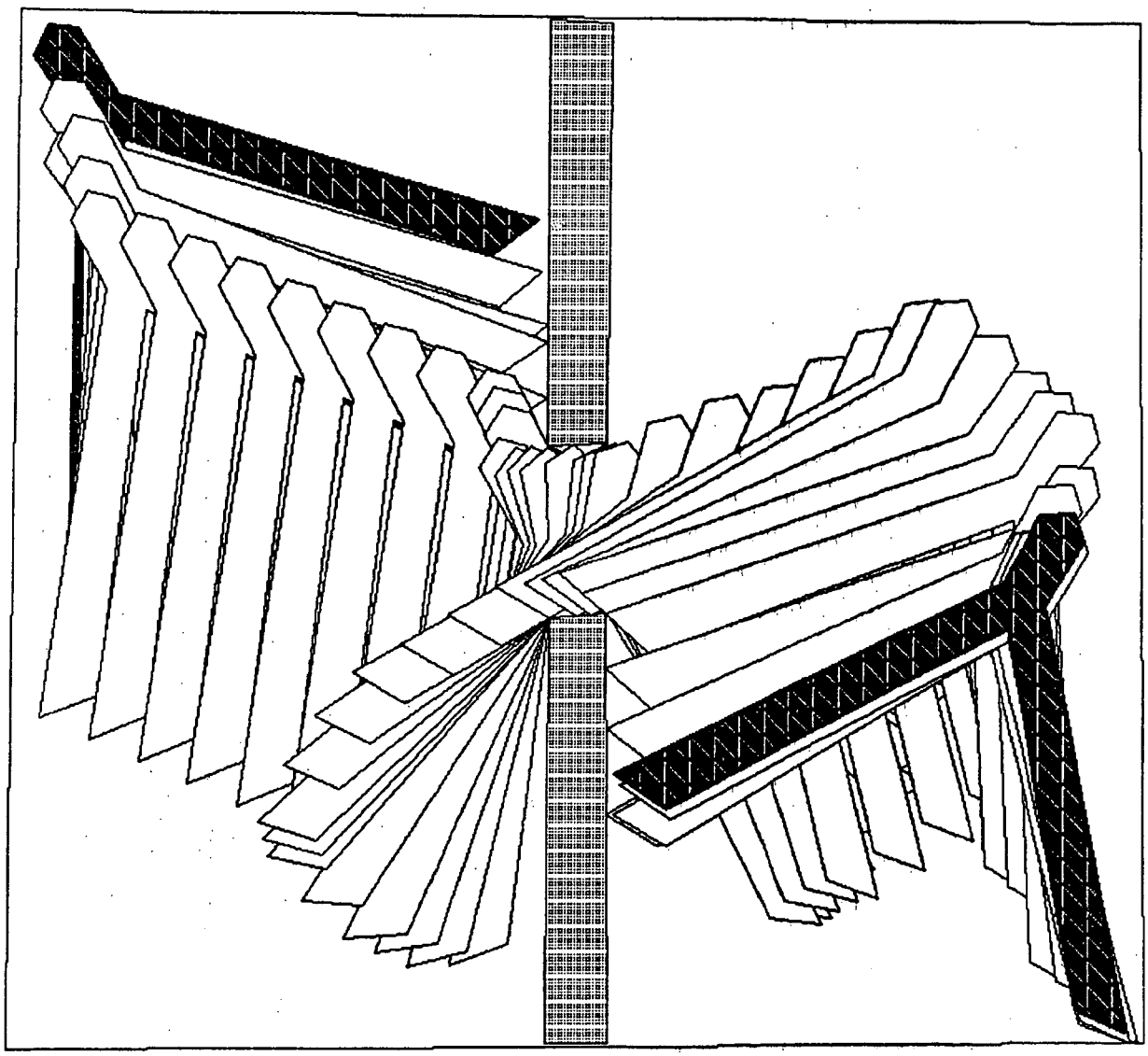
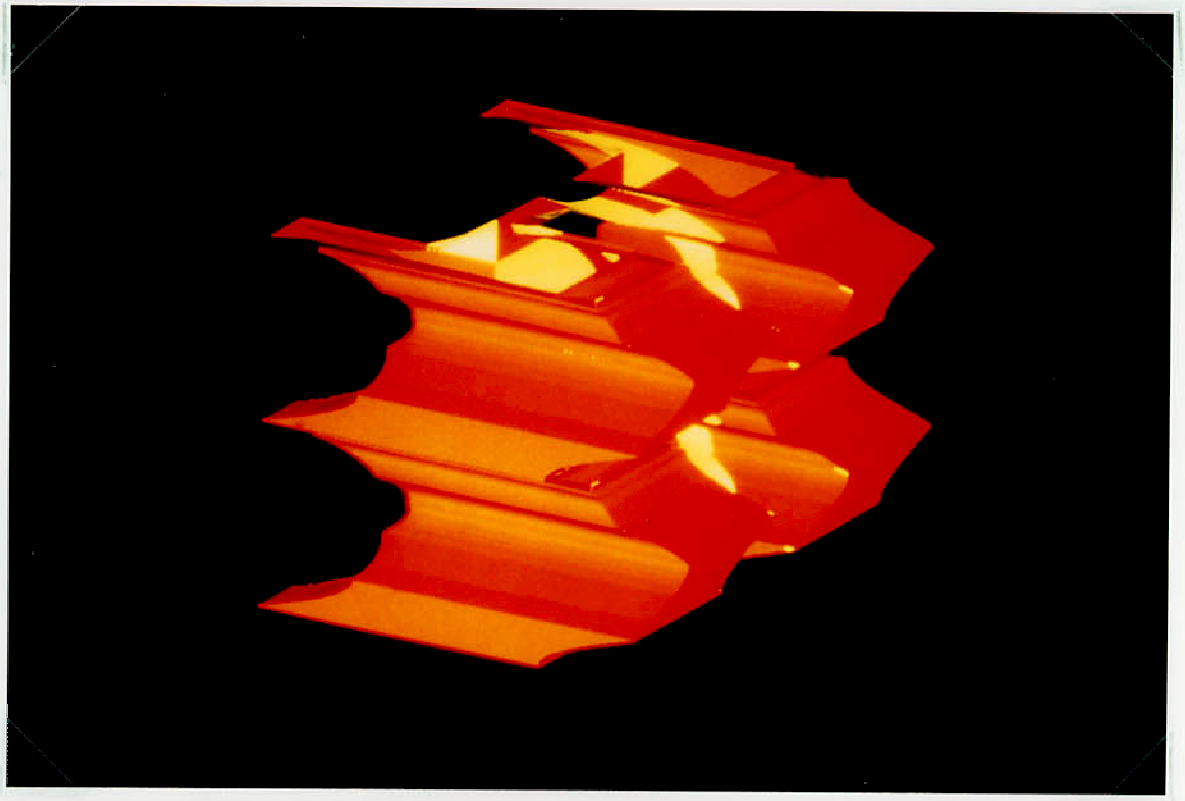
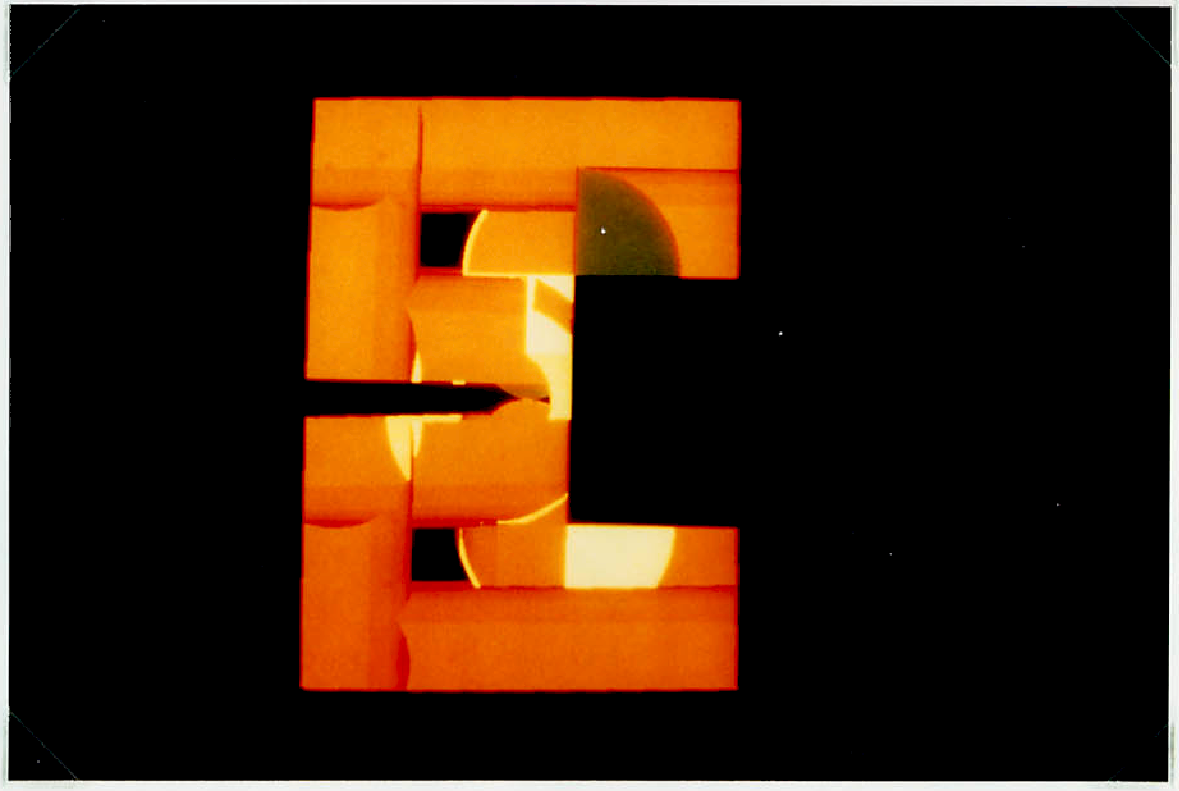


Figure 6a: Free space (perspective and top view), free motion





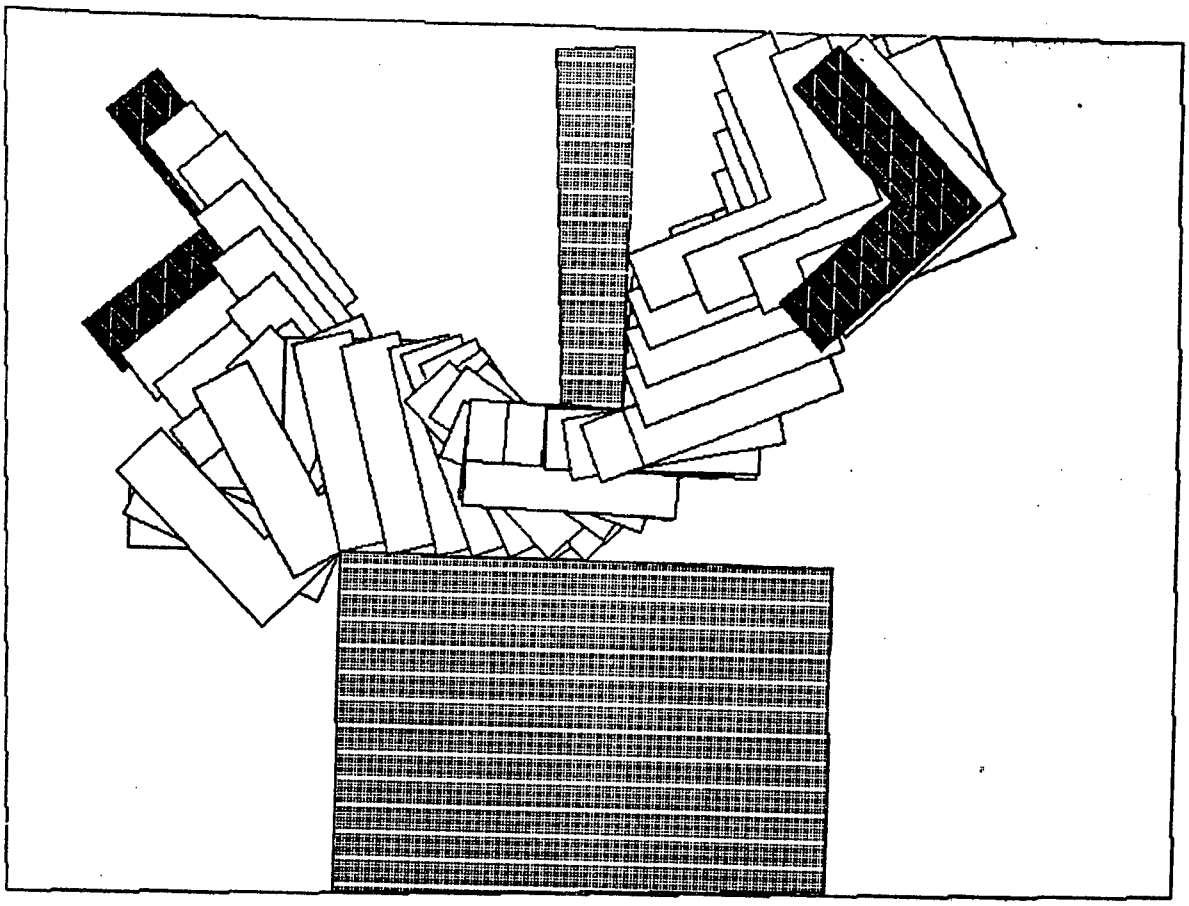


Figure 6b: Free space (perspective and top view), free motion

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique