May 1989 Report No. STAN-CS-89-1257 2

AD-A209 890

Robot Motion Planning: A Distributed Representation Approach

by

Jerome Barraquand and Jean-Claude Latombe

Department of Computer Science

Stanford University

Stanford, California 94305

LELAND STANFORD JUNIOR UNIVERSITY ORGANIZED 1891

89 6 26 065

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE		Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) STAN-CS-89-1257	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Stanford University; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Stanford, CA 94305	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION DARPA ONR; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAA21-89-C-0002 N00014-88-K-0620
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Robot Motion Planning: A Distributed Representation Approach

12 PERSONAL AUTHOR(S) Jerome Barraquand and Jean-Claude Latombe

13a TYPE OF REPORT research	13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day) 89-5-22	15 PAGE COUNT 54

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

In this paper, we propose a new approach for planning the motion of robotic systems among obstacles, which is based on a distributed representation of the world model. Within this approach, we designed and implemented a general purpose path planner with five new capabilities:

(1) It is able to generate very complex motions for robots with many degrees of freedom. In particular, we succeeded in generating complex paths for a 10 DOF non-serial manipulator arm made with both revolute and prismatic joints.

(2) It is drastically faster (between 1 and 2 orders of magnitude) than existing systems on a sequential computer. We generated complex paths for a 3 DOF bar in a 2D workspace in about 1 second on a MIPS-based workstation, as opposed to minutes or even tens of minutes for other algorithms.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Jean-Claude Latombe	22b TELEPHONE (Include Area Code) 415-723-0350; 22c OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

(3) The algorithms are highly parallelizable. We envision a VLSI implementation which should allow to generate plans in real-time, even for 10 DOF arms. Real-time motion planning for complex robotics systems opens new perspectives on some key issues such as motion planning in incompletely known or unknown environment, motion planning among fast moving obstacles and multi-robot motion planning.

(4) Our approach extends to motion planning problems with non-integrable constraints, e.g. non-holonomic and/or dynamic constraints. In particular, the planner was able to deal systematically with maneuvers for a non-holonomic car.

(5) The planner outputs a path for the robot in configuration space, while the goal is specified in operational space. Hence, the inverse kinematic problem is completely avoided. Furthermore, any kind of redundancy of the robot arms can be handled without modification. [illegible]

These capabilities are obtained through the systematic use of low-level distributed multiscale representations of the world. These representations allow to apply simple and powerful numerical techniques to geometrical and physical concepts.

The approach is illustrated by several simulation examples, both with mobile robots and manipulator arms.

Robot Motion Planning: A Distributed Representation Approach

Jerome Barraquand, Jean-Claude Latombe

Robotics Laboratory
Computer Science Department
Stanford University

Abstract

In this paper, we propose a new approach for planning the motion of robotic systems among obstacles, which is based on a distributed representation of the world model. Within this approach, we designed and implemented a general purpose path planner with five new capabilities:

(1) It is able to generate very complex motions for robots with many degrees of freedom. In particular, we succeeded in generating complex paths for a 10 DOF non-serial manipulator arm made with both revolute and prismatic joints.

(2) It is drastically faster (between 1 and 2 orders of magnitude) than existing systems on a sequential computer. We generated complex paths for a 3 DOF bar in a 2D workspace in about 1 second on a MIPS-based workstation, as opposed to minutes or even tens of minutes for other algorithms.

(3) The algorithms are highly parallelizable. We envision a VLSI implementation which should allow to generate plans in real-time, even for 10 DOF arms. Real-time motion planning for complex robotics systems opens new perspectives on some key issues such as motion planning in incompletely known or unknown environment, motion planning among fast moving obstacles and multi-robot motion planning.

(4) Our approach extends to motion planning problems with non-integrable constraints, e.g. non-holonomic and/or dynamic constraints. In particular, the planner was able to deal systematically with maneuvers for a non-holonomic car.

(5) The planner outputs a path for the robot in configuration space, while the goal is specified in operational space. Hence, the inverse kinematic problem is completely avoided. Furthermore, any kind of redundancy of the robot arms can be handled without modification.

These capabilities are obtained through the systematic use of low-level distributed multiscale representations of the world. These representations allow to apply simple and powerful numerical techniques to geometrical and physical concepts.

The approach is illustrated by several simulation examples, both with mobile robots and manipulator arms.

ACKNOWLEDGMENTS: *This research was funded by* DARPA contract DAAA21-89-C0002 (Army), DARPA contract N00014-88-K-0620 (Office of Naval Research), SIMA (Stanford Institute of Manufacturing and Automation), CIFE (Center for Integrated Facility Engineering), and a Digital Equipment Corporation grant.

The authors also thank Professor M. Harrison from the Stanford University Graduate Business School and J. Cheng from the Stanford University Statistical Consulting Service for their sound advice on Brownian motions with reflective boundaries.

1 Introduction

In this paper, we propose a new approach for planning the motion of robotic systems among obstacles, which is based on a distributed representation of the world model. Within this approach, we designed and implemented a general purpose path planner with five new capabilities:

(1) It is able to generate very complex motions for robots with many degrees of freedom. In particular, we succeeded in generating complex paths for a 10 DOF non-serial manipulator arm made with both revolute and prismatic joints.

(2) It is drastically faster (between 1 and 2 orders of magnitude) than existing systems on a sequential computer. We generated complex paths for a 3 DOF bar in a 2D workspace in about 1 second on a MIPS-based workstation, as opposed to minutes or even tens of minutes for other algorithms.

(3) The algorithms are highly parallelizable. We envision a VLSI implementation which should allow to generate plans in real-time, even for 10 DOF arms. Real-time motion planning for complex robotics systems opens new perspectives on some key issues such as motion planning in incompletely known or unknown environment, motion planning among fast moving obstacles and multi-robot motion planning.

(4) Our approach extends to motion planning problems with non-integrable constraints, e.g. non-holonomic and/or dynamic constraints. In particular, the planner was able to deal systematically with maneuvers for a non-holonomic car.

(5) The planner outputs a path for the robot in configuration space, while the goal is specified in operational space. Hence, the inverse kinematic problem is completely avoided. Furthermore, any kind of redundancy of the robot arms can be handled without modification.

These capabilities are obtained through the systematic use of a low-level distributed multiscale representation of the world. This representation allows us to apply simple and powerful numerical techniques to geometrical and physical concepts. More specifically, we compute numerical potential fields in configuration space that have very few or very small local minima, and we provide the planner with a way to escape systematically from these minima. The approach is illustrated by several simulation examples, both with mobile robots and manipulator arms.

The principle of our approach is to throw attractive potential fields over the workspace, each applying to a specific point on the robot body, and then to combine these potentials in configuration space to attract the whole robot toward the desired goal. Each of the workspace potential fields is computed numerically and has no other local minimum than the goal of the robot's point it acts on. Basically, these potentials allow to avoid all the *workspace concavities*. However, when these potentials are applied concurrently at the different points of the robot, the resulting potential in configuration space may have (and indeed has) local minima other than the goal. However, these minima generally determine wells of relatively small depth. The idea then is to build a graph connecting the local minima and perform a search of this graph until the goal is attained. The search of the local minima graph can be implemented in several ways.

For example, [Barra 89] presents an approach that connects the different local minima by tracking the *valleys* of the potential. This approach worked pretty well for a small number of DOF, but appeared unable – at least with the current implementation – to deal with some complex motions for a large number of DOF, although it succeded in some cases.

In this paper, we present two new approaches for building and searching this local minima graph. The first consists of a simple brute force exploration of the local minima in the discretized configuration space, which happens to be very efficient when applied to robots with a small number of DOF. The second is based on a Monte-Carlo procedure. It is much more general (it has solved tricky path planning problems for a 10 DOF robot), and furthermore highly parallelizable. We strongly believe that this second approach is definitely the best when massively parallel hardware becomes commonly available. However, the first method is faster for small robots on a sequential computer and is *deterministically resolution-complete*, whereas the Monte-Carlo approach is only *probabilistically resolution-complete*.

We model the obstacles as distributed bitmap descriptions rather than centralized semi-algebraic descriptions as it is usually the case in the literature on path planning. We think that distributed descriptions are simpler to obtain from sensory data and that they may be easier to handle algorithmically (specially when massively parallel computing hardware becomes commonly available). In particular, the 'perfect potential fields' computed numerically in our approach could not in general be computed analytically with closed mathematical functions. In addition, using bitmap descriptions opens new perspectives on some computational complexity issues. While the complexity of path planning algorithms using semi-algebraic models is usually polynomial in the number of algebraic constraints and in the maximal degree of these constraints, the complexity of our algorithm is polynomial in the inverse of the resolution of the bitmap descriptions[1].

We have implemented our approach and experimented with it on many examples, simulating both mobile robots and manipulator arms. Some of the most significant experiments are reported in this paper. In particular, for a 3 DOF bar amidst a planar maze composed of more than 70 complex-shaped obstacles, we have succeeded in generating paths in about 5 second on a DEC-3100 workstation. We have also been able to generate complex paths including several maneuvers for a non-holonomic 3 DOF planar mobile robot (a car with limited steering angle) in about 1 minute. To our knowledge, our planner is the first one capable of solving systematically non-holonomic motion planning problems. Finally, and most importantly, we have succeeded in generating complex paths for a non-serial 10-DOF manipulator arm including both prismatic and revolute joints. All the examples presented

[1]Different algorithm complexity measures with non algebraic models have already been given in [Lumelsky 87].

so far to the planner have been solved within a few minutes of computation. However, as the Monte-Carlo procedure is not deterministically complete, and as the path planning problem is known to be NP-Hard, it is certainly possible to find problems which the algorithm would fail to solve in a reasonable amount of time. We are currently investigating this issue. To our knowledge, our planner is the first one capable of solving systematically motion planning problems of this complexity.

As the Monte-Carlo procedure is easily parallelizable, we envision a VLSI implementation of the algorithm for 10 DOF manipulator arms, which should allow to generate plans in real-time. If motion planning can be performed in real-time, it can be included in closed loop in control algorithms. The concept of 'reactive planning' then reduces to closed-loop real-time motion planning. Therefore, real-time motion planning for complex robotics systems opens new perspectives on some key issues in robotics such as motion planning in incompletely known or unknown environment, motion planning among fast moving obstacles, multi-robot motion planning.

The paper is organized as follows. In Section 3, we describe how the workspace numerical potential fields are computed and how a generic robot interacts with these potentials. In Section 4, we detail the very fast hierarchical graph search planner implemented for 3 DOF, and we show some experimental results obtained with a long bar in different 2D workspaces. We also compare these results with those computed using the Monte-Carlo procedure. In Section 5, we describe the application of the technique to *non-holonomic constraints*. We show some experimental results where a car with limited steering angle plans complex maneuvers. In Section 6, we give a detailed account of the Monte-Carlo approach which allowed to solve complex planning problems for complex robots (in particular for the 10 DOF non-serial manipulator arm) and show corresponding simulation results. Finally, we discuss in Section 7 the parallelization issues relative to the different steps of the motion planning algorithm. All the experiments reported below were carried out on a MIPS-based DEC-3100 workstation.

2 Relation to other work

Research on motion planning has been quite intensive during the past ten years. Most of it has focused on path planning, i.e. the topological and geometrical problem of finding a collision-free path between two given configurations of a robot, or more generally between two given subsets of configuration space. Today, the mathematical and computational structures of the general problem (when stated in algebraic terms) is reasonably well understood ([Schwartz 82,Schwartz 87a,Schwartz 87b,Canny 87]). In addition, some practical algorithms have been implemented in more or less specific cases (e.g., [Brooks 83,Fav 84, Gouzenes 84,Lozano 87]).

One of the most extensively studied path planning approach is the cell decomposition approach. It consists of first decomposing (exactly or approximately) the set of free configurations of the robot into a finite collection of cells (e.g., Collins cells for exact decomposition and rectangular cells for approximate decomposition), and then searching a graph representing the connectivity between these cells. However, in this approach, the number of cells to be generated is a function of the number of semi-algebraic constraints used to model the robot and the obstacles, and of the degree of these constraints. This function tends to grow exponentially with the number n of DOF, as the volume of the configuration space (locally diffeomorphic to $\mathbf{R}^n$) increases exponentially with n. Thus, the approach is untractable even for reasonably small values of n. To our knowledge, no planner has been implemented using this approach with $n > 4$. In fact, this is true of other so-called 'global' methods, which represent the connectivity of free space as a graph before actually starting the search for a path.

In an attempt to cope with more DOF, an approach, which consists of approximating free space at successive levels of detail as a collection of rectangular cells of decreasing size, has been developed by several researchers (see [Brooks 83,Fav 84]). Several implementations of this approach show that it can significantly improve the average computing time for small numbers of DOF (typically, $n \leq 4$). However, although the full potential of the paradigm has probably not been completely exploited yet, current results lead to think that it is not powerful enough to extend the applicability of the cell decomposition approach to significantly higher values of n.

Since the untractability of the cell decomposition approach – and more generally of the other global methods – is due in part to the precomputation performed before the search of the connectivity graph, local methods to path planning have been considered for handling more DOF and some successful systems have been implemented (e.g., [Donald 84,Fav 87]). A local path planning method consists of placing a regular grid (at some resolution) onto the robot configuration space and searching this grid (see [Donald 87]). Heuristics computed from partial information about the geometry of the configuration space are used to guide the search. Thus, unlike global methods developed so far, a local method requires no expensive precomputation step before starting the search of a path. Consequently, in favorable cases, it runs substantially faster than any global method. But, since the search graph (i.e., the grid) is considerably larger than the connectivity graph searched by global methods, in less favorable cases, it may require much more time than global methods. In order to deal with this difficulty, powerful heuristics are needed to guide the search, but known such heuristics have the drawback of eventually guiding the search into dead-ends, from which it is very difficult to escape. For example, a widely used heuristic technique consists of guiding the robot along the gradient direction generated by an artificial potential field [Khatib 86]. It is well-known that this technique eventually leads the search to local minima of the potential and provides no way to escape these minima. The issue of defining a potential field with a unique minimum at the goal configuration and whose attraction domain includes all free space has been investigated with very limited success so far ([Kod 87,Rimon 88]).

Furthermore, if such a nice potential could be defined, its computation would probably be expensive and constitute a precomputation step before search similar in drawback to cell decomposition. Paths for a 8 DOF manipulator have been generated with a variant of the potential field method [Fav 87]. Although impressive, the results have been obtained in a specific workspace made of vertical cylindrical pipes. Such a workspace makes probably easier the definition of a potential field with few local minima. Nevertheless, the planner also required some human interaction for escaping local minima.

Recently, we have developed an approach based on a numerical valley tracking algorithm [Barra 89] which was able to plan paths for a 10 DOF manipulator arm with a complex kinematic chain (the same arm is used in this paper). However, the approach was not very reliable - it failed on several problems - and quite slow. In fact, the approach reported below derives from that described in [Barra 89].

3 Distributed Representation and Potential Fields

3.1 Overview

Let $\mathcal{A}$ denote the robot body, $\mathcal{W}$ its workspace, and $\mathcal{C}$ its configuration space parameterization (i.e. generalized coordinates). A configuration of the robot, i.e. a point in $\mathcal{C}$, completely specifies the position of every point in $\mathcal{A}$ with respect to a coordinate system attached to $\mathcal{W}$. The subset of $\mathcal{C}$ consisting of all the configurations where the robot has no contact or intersection with the obstacles in $\mathcal{W}$ is called free space and denoted $\mathcal{C}_{free}$.

The workspace $\mathcal{W}$ is modeled as a multiscale pyramid of bitmap arrays, each of these arrays being 2- or 3-dimensional. At a given resolution level, the array is represented by a function:

$$BM : \mathcal{W} \rightarrow \{1,0\}$$

$$x \mapsto BM(x)$$

in such a way that the subset of points x such that $BM(x) = 1$ represents the workspace obstacles and the subset of points x such that $BM(x) = 0$ represents the empty part of the workspace. All our experiments were conducted in a 2D workspace, but the approach extends without any change two a 3D workspace (at the expense of more computing time).

The coarsest level of the bitmap pyramid is typically 16^2, the finest 256^2 or 512^2,

The bitmap representation of a particular workspace at the 256^2 resolution is shown in figure 1 (1 = black; 0 = white).

For each point p in $\mathcal{A}$, one can consider the geometrical relation which maps the configurations $q = (q_1, \ldots, q_n)$ of the robot to the position x of p in the workspace. This geometrical map:

$$X: \ \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{W}$$

$$(p, q) \mapsto X(p, q) = x$$

is called *forward kinematic relation.*

Our approach to path planning is based on the definition of a numerical potential field $U(q)$ over configuration space that has few and small local minima. In the case where some local minima appear, the algorithm backtracks in configuration space until the robot gets out of the local minimum. All the computations, in configuration space as well as in workspace, are performed within a multiscale hierarchical data representation. For each resolution level of the hierarchy, the overall configuration space potential field $U(q)$ is computed as a non-linear combination $U(q) = G(U_{p_1}(X(p_1, q)), \ldots, U_{p_k}(X(p_k, q)))$ of several workspace potential fields $U_{p_1}(x), \ldots, U_{p_k}(x)$, each of these attracting concurrently a given point p_i of the robot towards its goal position. In the next two subsections, we describe the computation of these workspace potential fields. Then, in subsection 3.4, with describe the interaction of the robot with these workspace potentials, and specify the non-linear combination G.

3.2 Distance Function and Voronoi Diagram

At each resolution level of the hierarchy, the potential field used in our implementation is constructed in two steps. In the first step, we compute an approximation to the distance to the obstacles in the workspace, as mentioned in [Barra 89]. We build at the same time an approximation of the generalized Voronoi Diagram. This first step is described in this subsection. In the second step, we compute a potential field using this distance information and the Voronoi diagram. This second step is described in the next subsection.

We first compute the map d_1, which is an approximation of the L^1 distance to the obstacles. The function d_1 is simply computed as follows: First the points on the boundary of $\{x \ / \ BM(x)\}$ are identified and the value of d_1 at these points is set to zero (we also include the points on the rectangular frame bounding the bitmap as boundary points). Then, the value of d_1 at all the neighbors of these boundary points in the empty part of the workspace are set to 1; the value of d_1 at the neighbors of these points, if not yet computed, is set to 2; etc. The procedure is recursively repeated until all the empty part of the workspace is completely explored. Figure 2 displays the equipotentials of d_1 for the workspace shown at figure 1. The valleys of the map thus obtained, i.e. the set of points where the waves coming from the boundary points meet, form the discretized Voronoi diagram for the L^1 distance.

As a matter of fact, we can define the generalized Voronoi diagram associated with any distance function in the following way: For each point x in the free space, we can compute along with the distance d_1 to the nearest obstacle the set $P(x)$ of boundary points from

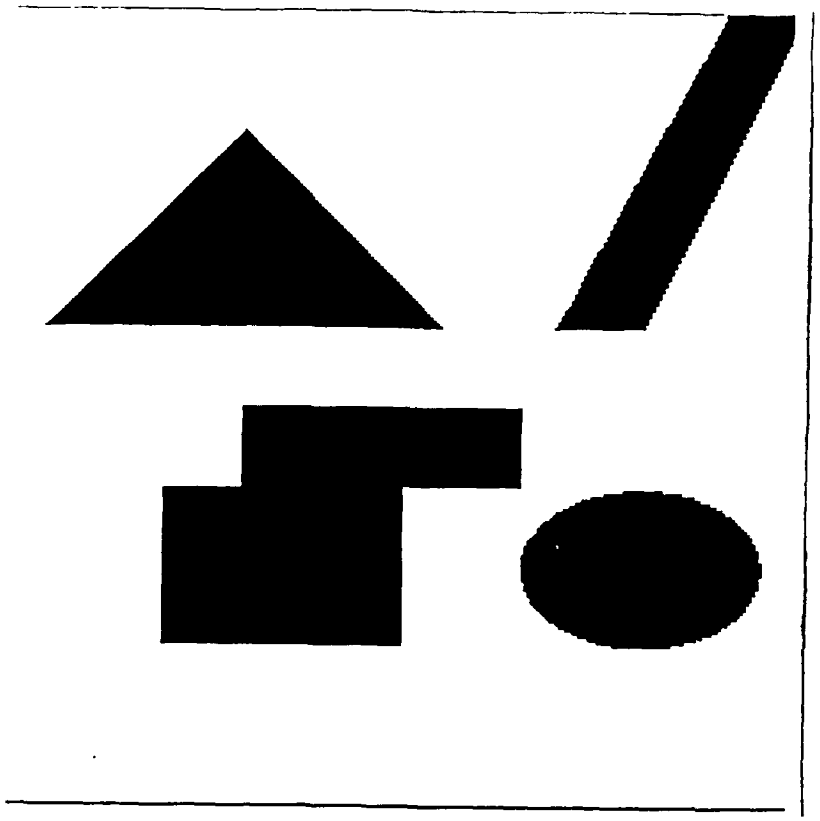

Figure 1: Bitmap description of a 2D workspace.

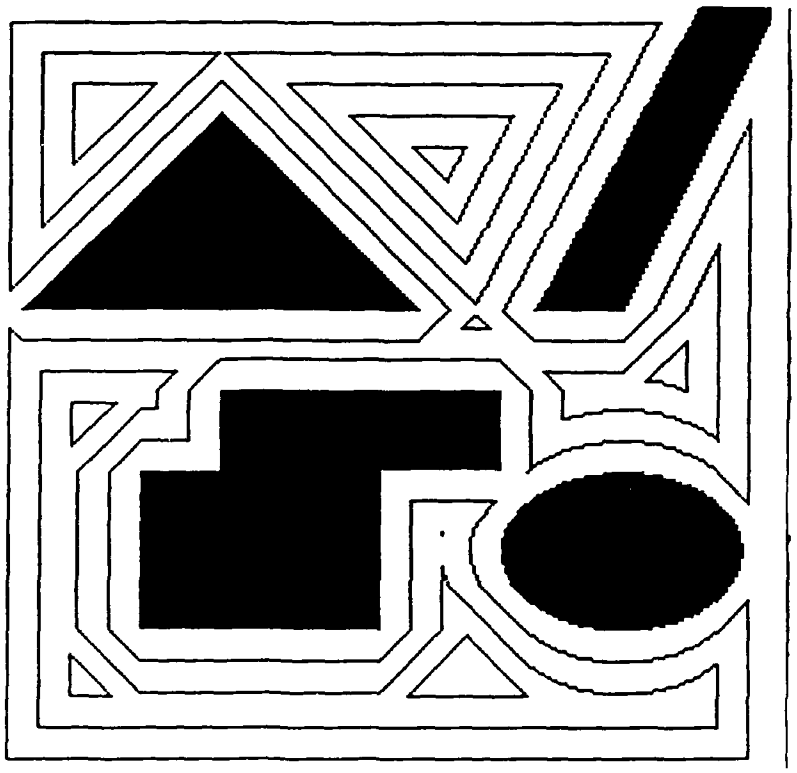

Figure 2: Visualization of d_1 for the 2D workspace shown above.

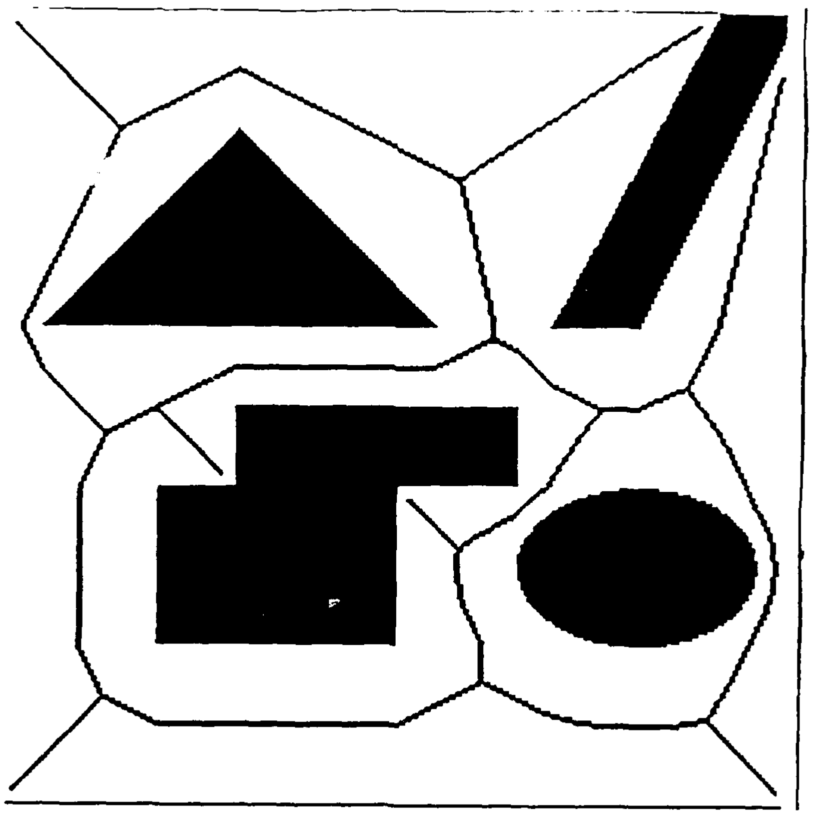

Figure 3: A Generalized Voronoi Diagram.

which this distance has been obtained. This associated set of boundary points $P(x)$ is the set of points which realize the minimum distance $d_1(x)$. The generalized Voronoi diagram is precisely the set of points for which $P(x)$ contains more than one boundary point. Figure 3 displays the generalized Voronoi diagram computed from the workspace shown on Figure 1.

The computation of d_1 is not local and therefore must be done prior to the execution of the rest of the path planning algorithm. However, the procedure is linear in the number of points of the bitmap representing $\mathcal{W}$ and is quite fast (a fraction of a second for a 256^2 bitmap). The time complexity of the procedure does not depend on the shape of the obstacles (the complexity of computing analytically the Voronoi diagram of a polygonal workspace increases with the number of vertices).

3.3 Workspace Potential Fields Without Local Minima

The idea is to define an attractive potential field [Khatib 86] attracting some points of the robot $\mathcal{A}$ toward their goal position. We could use a classical analytical potential field depending on the Euclidean distance in $\mathcal{W}$. But it is known that this kind of field can have other local minima than the goal, especially in the workspace obstacle concavities. The bitmap description of the workspace allows us to compute numerically an attractive potential field which has no other local minima than the goal, when used on a point robot. This potential happens to be very helpful for avoiding concavities of the workspace obstacles.

Let us consider a point robot x moving freely in the workspace. Let x_{GOAL} be the goal

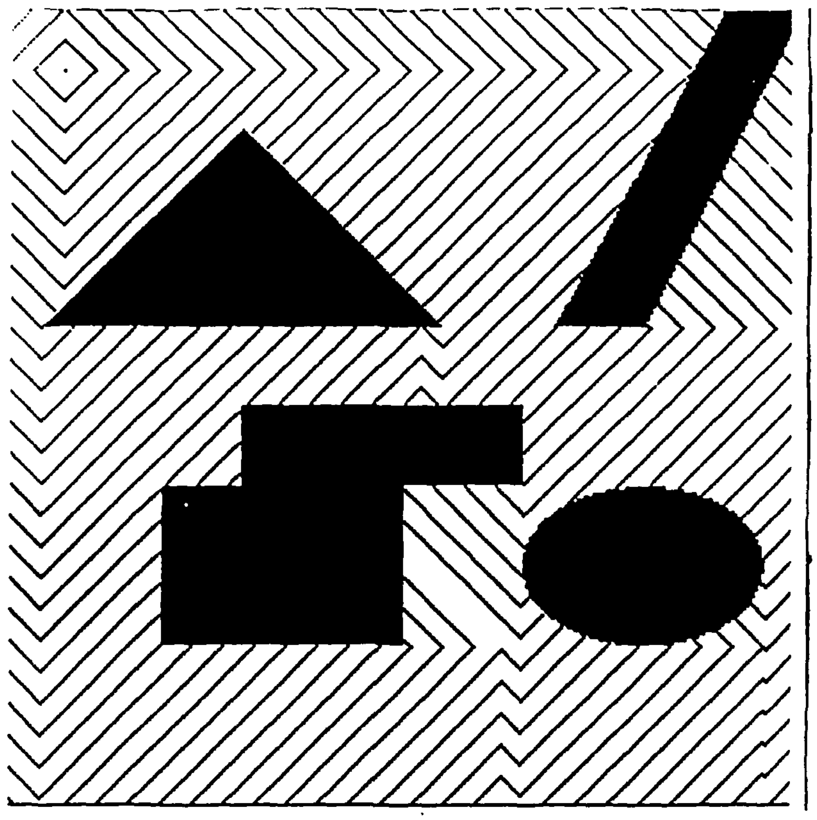

Figure 4: Numerical potential field U_W for a 2D workspace.

position of x in $\mathcal{W}$. We set the value of the potential at the goal point to zero. Then the neighbors of x_{GOAL} are set to 1 . This procedure is recursively repeated until the empty part of $\mathcal{W}$ is completely explored. The complexity is linear in the number of points of the bitmap description, and constant in the number and shape of obstacles. The equipotentials of the resulting workspace potential field $U_{\mathcal{W}}(x, x_{GOAL})$ for the 2D workspace of figure 1 are displayed in figure 4. This computation was performed in a fraction of a second.

A property of this function is the following: when one follows the gradient from any starting point x_{INIT}, a path connecting x_{INIT} to x_{GOAL} is obtained, which is the shortest geometrical path for the L^1 distance. In a 3D space, this may be an important advantage over exact shortest distance algorithms, since the problem of computing the exact shortest distance in a 3D polyhedral space is NP-hard in the number of vertices under any L^p metric [Canny 87].

The above technique has interesting extensions. For example, it is possible to modify the definition of the potential in order to take safety considerations into account.

We detail in the next paragraph an improvement of the technique previously presented in [Barra 89] for building 'safe' potential fields. The idea is to compute first the distance d_1 and the Voronoi diagram V_D as explained above. Then, the potential field is computed in three steps. The first step consists of tracing a line L following the gradient of the distance d_1 from the goal point x_{GOAL} to the nearest point on the V_D. Once this line is computed, we obtain a new set of points $S = V_D \cup L$. This set S is connected. The second step of the algorithm is to label all the points of S starting from x_{GOAL}. The label 0 is assigned

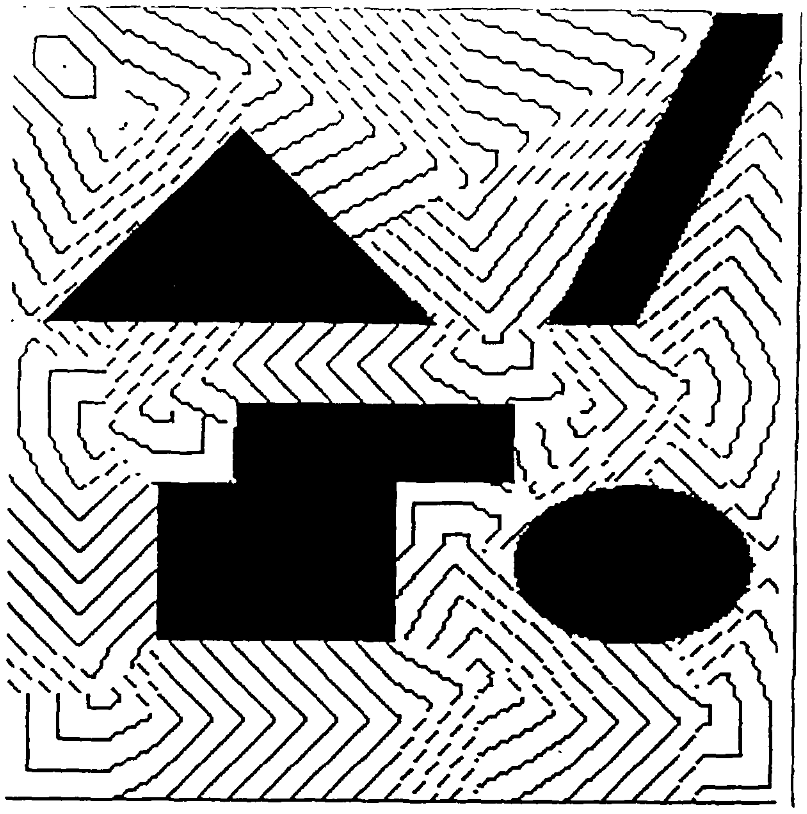

Figure 5: Another numerical potential field taking safety considerations into account.

to x_{GOAL}, the label 1 is assigned to its neighbor x_1 which is at maximum distance d_1 of the obstacles. Then we consider together the neighbors of x_{GOAL} and x_1 which have not been labeled yet, and give the label 2 to the one which is the farthest from the obstacles. This operation is recursively repeated until the set $S = V_D \cup L$ is completely explored. At each step of the recursion, the list of points in S to be labeled is represented with a heap structure, so that each insertion of a new point or extraction of the maximum distance point is performed in logarithmic time. At the end of the second step, all the points $x \in S$ have a label $l(x)$. The third step is simpler, and very similar to the computation of d_1. We compute all the neighbors of all the points in S, and give to the neighbor of $x \in S$ the label $l(x) + 1$. We then compute the neighbors of these neighbors, and increment the labels iteratively until the freespace is completely explored.

Instead of incrementing the labels systematically of the same constant 1, more complex incrementation functions giving smoother potentials can be easily derived. For example, we can make the increments in the third step dependent of the distance d_1. Instead of giving the increment 1 to the neighbors of S, we could give the increment $1/d_1$. Then, we could give the increment $1/d_1$ again to the neighbors of these neighbors, and so on. In that case, we would obtain a potential field that becomes infinite on the obstacles boundaries. We will not detail further the cosmetics of numerical potential fields computations, but we want to point out that a large familly of potential fields with very nice properties can be build within this bitmap numerical framework.

Figure 5 shows the equipotentials of this last potential field. The path obtained by tracking its gradient stays away from the obstacles (in fact a point robot tracking this potential would follow the safest path on the Voronoi diagram.). However, the complexity of computing this modified potential is slightly higher than that of the previous one. Let m be the number of points in the workspace bitmap, and p the number of points lying in $V_D \cup L$. The complexity is $O(m + p \log p)$ instead of $O(m)$ (the logarithmic term comes from the calculation of the maximum distance point in the Voronoi list with the heap structure). For most reasonable workspaces, we have $p \propto m^{\frac{n-1}{n}}$ because the Voronoi diagram is of dimension $n - 1$ in a n-dimensional workspace. Therefore, we have an average complexity of $O(m + m^{\frac{n-1}{n}} \log m)$, which simply equals $O(m)$. The algorithm is still linear in the average case. For a 256^2 workspace, the computation took about 2 seconds (including the computation of d_1 and V_D).

3.4 Configuration Space Potential Field

As the workspace representation is distributed, the natural representation for the robot should also be distributed. That is to say, the robot should be considered as a the discretized collection of all the points on $\mathcal{A}$. On a parallel hardware, this would be the simplest and most efficient fashion to compute the distance of the robot to the obstacles and the attractive potentials. One could use one processor per discretized point on the robot. However, as we made all our simulations on a sequential computer, we slightly adapted this representation in order to reduce the amount of computations.

First, we only need to consider the points on the boundary of the robot, because the minimum distance from the obstacles to the robot is always reached on the boundary of the robot. Second, using the distance information, we can adjust adaptively the number of points on the robot (we call them control points) with regards to the distance to the obstacles computed for each of these points. To illustrate this idea, let us assume that we want to compute the minimum distance D_1 from the robot to the obstacles. It is mathematically defined as:

$$D_1(q) = \inf_{p \in \mathcal{A}} d_1(X(p, q)).$$

Let us consider an ideal robot composed of a single straight line segment of length L. We suppose that the workspace function d_1 has been precomputed. Instead of computing the distance d_1 for each discretized point on the segment and then calculating the minimum of all these distances, we choose to compute first the distances $d_1(begin)$ and $d_1(end)$ of the two extremity points of the segment. If $DD_1 = \min(d_1(begin), d_1(end)) > L$, then we are sure that the robot cannot hit the obstacles, and we choose DD_1 as an approximation of D_1. Otherwise, we compute the distance d_1 for the middle point of the segment. Our initial segment can be considered as the union of two half-length segments, and we can recursively apply the approximation procedure to these sub-segments.

If the robot boundary can be modeled by a polygon, this segment procedure can be applied to each segment of the polygon. Moreover, the segment procedure can be easily generalized to higher order models of the robot boundary like arcs of circles, arcs of ellipses, and so on. When the robot is far from the obstacles, very few points on its boundary need to be checked. When the robot is in contact space however, the segment which achieves the contact has to be checked in its entirety.

After this procedure, we obtain an approximation of the distance D_1 between the robot and the obstacles.

As our robot can be a complex articulated system, we might need to check its auto-intersections. For example, if the robot is a manipulator like those described in section 6.5, we have to check intersections of several straight line segments in the workspace. Usually, if the number of segments is n, checking the intersections of all these segments can be performed in $O(n^2)$ operations. With a distributed representation of the workspace, this complexity falls to $O(n)$: as a matter of fact, we can simply *draw* each of the segments on the workspace bitmap. If we draw a point which has already been drawn, we know that the segments intersect. This remark shows once again that distributed representations generate algorithmic complexity measures which are very different from those obtained with algebraic representations.

We now detail the computation of the configuration space potential field for a generic robot.

Let us first consider a simple bar in a 2D workspace like the one illustrated in figure 4.2. We want the bar to reach a given goal configuration from any initial configuration. To specify this goal configuration in terms of workspace potential fields applied to given points on the robot, selecting only one point will not be enough to characterize the goal. This is so because one point on the robot specifies only two degrees of freedom, whereas our robot has three. Therefore, we need to select at least two points on the robot if we want to specify completely the final configuration. For more complex robots like manipulator arms with many DOF, one might want to specify only some points on the end-effector, in order to define a given grasping configuration without computing all the inverse kinematics. So, depending on the application considered, the choice and the number of the attractive points for defining the overall potential field can be conveniently adapted.

Let us call k this number of attractive points, and $p_1, \ldots, p_k$ the corresponding points on the robot body.

The overall configuration space potential field $U(q)$ is computed as a combination $U(q) = G(U_{p_1}(X(p_1,q)), \ldots, U_{p_k}(X(p_k,q)))$ of several workspace potential fields $U_{p_1}(x), \ldots, U_{p_k}(x)$, each of these attracting concurrently a given point p_i of the robot towards its goal position.

To define precisely this overall potential, we still need to specify the competition function G. In most of the previous works on the artificial potential field approach, the function G was chosen linear [Khatib 86]:

$$G(y_1, \ldots, y_k) = \sum_{i=1}^{i=k} \lambda_i y_i$$

This choice gives good results in some cases, but the identification of the multipliers λ_i is a non-trivial problem.

The choice of the function G is especially important because it highly influences the number of local minima of the potential U. With our 'perfect' numerical potential fields defined for a point robot, the workspace concavities do not create any more local minima by themselves. It is the concurrent attraction of the different points on the robot which creates these local minima, because of the fact that these points do not move independently. The function G precisely defines in which way the competition between the different points is going to be regulated.

The choice of G which seems to minimize experimentally the number of local minima is

$$G(y_1, \ldots, y_k) = \min_{i=1}^{i=k} y_i$$

This competition function favors the attraction of the point which is already in the best position to reach its goal. It seems natural to think that this policy will minimize the number of competitions between points, and therefore minimize the number of local minima. However, when one point has reached its goal position, the potential field is identically null, and the attraction paradigm does not work any more to bring the other points to their goal position. A solution to avoid this phenomenon is to add another term to the competition function:

$$G(y_1, \ldots, y_k) = \min_{i=1}^{i=k} y_i + \epsilon \max_{i=1}^{i=k} y_i$$

where ϵ is a small number. Then, we have to tune the parameter ϵ. In the case of the bar in a 2D workspace illustrated on figure 4.2, the best results where obtained with $\epsilon = 0.1$.

Another choice is to set:

$$G(y_1, \ldots, y_k) = \max_{i=1}^{i=k} y_i$$

This choice would tend to maximize the number of competitions between points, and therefore the number of local minima. However, it can be the best choice in some cases, and especially with manipulator arms. As a matter of fact, the number of local minima is not the only measure for the quality of the overall potential. Another very important factor for the convergence rate of the algorithm is the distribution of depths of these local minima. The method will overcome small local minima without problem even if they are numerous. But only one very deep local minimum can provoke the failure of the whole algorithm. Another word for the depth of a local minimum is the *activation energy*, to refer to chemical kinetics terminology. This last choice of the competition function seems to maximize the number of local minima but to minimize the activation energy for each of these. It is the one we chose for the 10 DOF manipulator arm illustrated in figure 6.5. In our experiments, we chose two attracting points at the two end-effectors of the non-serial kinematic chain.

It may appear surprising to choose such competition functions involving computations of maxima and minima. These are non-differentiable for some degenerate points, and computing their derivatives may cause numerical instabilities. However, there are many techniques for following the gradient of a function, some of which are faster and fancier, some of which are slower and more robust. In all cases, one can design a differentiable approximation of the maximum function in the following way: Choose a very small number σ, and compute:

$$G(y_1,\ldots,y_k) = \sigma \log \left(\sum_{i=1}^{i=k} \exp(y_i/\sigma) \right)$$

One can show that this function uniformly converges towards the maximum function when σ converges towards zero.

The next section describes the first of the two paradigm we have used to escape from local minima. It is valid only for a small number of DOF, but happens to be the fastest on a sequential computer for robots with 3 DOF. A general paradigm valid for any number of DOF will be presented in section 6.

4 Very Fast Motion Planning for a Small Number of DOF

4.1 Hierarchical Best First Search of Freespace

We represent C as a cartesian n-dimensional space with modular arithmetic for the angles. For example, in the case of the 3 DOF mobile robot studied in section 4.2, the configuration space is represented by $\mathbf{R}^2 \times \mathbf{R}/2\pi Z$. We then perform a hierarchical discretization of the cartesian n-dimensional space using a multiresolution pyramid. The resolution level chosed in the configuration space pyramid is tightly related to the resolution level chosen in the workspace bitmap representation. To make this relation precise, we need to give some elementary definitions related to the concept of multiresolution.

Let us denote δ the effective distance between two adjacent points in the workspace bitmap representation. In the workspace pyramid, δ varies between δ_{min} and δ_{max}.

For example, let us assume that we have a 2D workspace represented by a pyramid of images whose sizes are ranging between 16^2 and 512^2. If the distance is measured in percentage of the workspace diameter, we will have in that case $\delta_{min} = 1/512$ and $\delta_{max} = 1/16$. The scaling factor between two successive resolutions is always 2 in our implementations, although it could be any other factor. The resolution is by definition the logarithm in the base defined by the scaling factor of the inverse of the distance between two discretization points. In our example, the resolution r varies between $r_{min} = -\log_2(\delta_{max}) = 4$ and $r_{max} = -\log_2(\delta_{min}) = 9$.

For any given resolution chosen in the workspace $\mathcal{W} \subset \mathbf{R}^d$, say $r = -\log_2(\delta)$, the corresponding resolution $R_i = -\log_2(\Delta_i)$ for each of the degrees of freedom q_i must be chosen in such a way that any elementary motion (i.e. a motion of $\Delta_i = 2^{-R_i}$) in configuration space is transformed in a small motion of all the robot body points in workspace. By small motion, we mean a motion for which any point on the robot moves by less than a few number *nbtol* points in workspace.

The relation between positions in workspace and configurations is given by the forward kinematics relation $X(p,q)$. So, an elementary motion Δ_i of q_i in $\mathcal{C}$ will generate a motion of each coordinate X_j in workspace for each point p on the robot body of:

$$\frac{\partial X_j}{\partial q_i}(p,q)\Delta_i$$

If we impose all workspace motions to be less than $nbtol \times \delta$, we must have:

$$\Delta_i = nbtol \times \delta / \sup_{p\in\mathcal{A}, q\in\mathcal{C}, j\in[1,d]} \left(\frac{\partial X_j}{\partial q_i}(p,q)\right) = nbtol \times \delta / J^i_{\text{sup}}$$

The numbers J^i_{sup} are generally straightforward to compute. Then the resolution R_i is computed by:

$$R_i = r + \log_2(J^i_{\text{sup}}) - \log_2(nbtol)$$

Let us consider as an example a bar of length L moving freely in a 2D workspace. The three degrees of freedom of this bar are respectively the first coordinate x_G of the center of gravity, the second coordinate y_G, and the orientation of the bar θ. x_G, y_G, and θ are all normalized coordinates ranging between 0 and 1.

One can compute

$$J^{x_G}_{sup} = J^{y_G}_{sup} = 1$$

and

$$J^{\theta}_{\text{sup}} = \pi L$$

If we set $nbtol = 2$, we have:

$$R_{x_G} = R_{y_G} = r - 1$$

and

$$R_\theta = r + \log_2(\pi L) - 1$$

This means that we need $2^1 = 2$ times less samples for x_G and y_G than for the workspace representation at each resolution, and $2/(\pi L)$ less samples for θ.

The uniform discretization of configuration space for each resolution level that we just described does not take advantage of the knowledge we have of the distance to the obstacles D_1. As a matter of fact, instead of moving with constant increments Δ_i we could compute for each configuration the largest increment possible, which would allow the robot to move fast, while still avoiding the obstacles.

As we know our distance D_1 to the obstacles, we can allow elementary motions in workspace of the order of D_1, instead of limiting ourselves to motions of the order of δ. Then, we can approximately estimate the configuration space increments in the following way:

$$\Delta_i(D_1) = nbtol \times D_1 / J^i_{\text{sup}}$$

Here, *nbtol* must be smaller than 1 in order to make sure that the obstacles are avoided. In cases where the workspace is not cluttered, this adaptation may reduce significantly the overall computation time. However, it makes the computation of the neighbors of a given point more complex than with the uniform grid.

The principle of the approach at each resolution is to perform a best first search of the configuration space discretized grid, using the potential field as the heuristic for the search. For a n DOF robot, each discretized configuration has $3^n - 1$ neighbors.

The graph search is first performed in the coarsest grid at resolution r_{min} in workspace and $R_i = r_{min} + \log_2(J^i_{\text{sup}}) - \log_2(nbtol)$ in configuration space. The workspace potential fields are also computed at the coarse resolution r_{min}. If no solution is found, the search is performed in the next finer grid, and so on until the finest grid is treated. At each resolution level considered, the potential fields have to be recomputed. As the computation time required to explore a coarser grid is almost neglectable with regards to the computation time required to explore a finer one, this naive coarse to fine approach is actually the best in the average case. It would be possible to imagine a hierarchical approach where the resulting graph of a given resolution could be used as the initial graph of the next finer resolution. However the solution of a path planning problem is generally so versatile with respects to the resolution that in most cases, the resulting graph of the previous resolution where the search algorithm has failed would be more misleading than helpful for the search in the next finer resolution.

The search algorithm in each grid uses the numerical potential field as the heuristic for a best-first procedure. As long as the algorithm does not reach a local minimum of the potential field, the best-first search reduces to following the gradient of the potential. When a local minimum is reached, the search algorithm fills up the attractor associated with this local minimum until a saddle point allows the best-first procedure to reach another attractor, and so on until the attractor of the goal is reached. At first sight, the experimental efficiency of such a brute force technique may appear fairly surprising. The fundamental reason of this efficiency is the following: The potential fields computed in the workspace are designed to be 'perfect' potential fields for a point robot, in the sense that they have no other local minima than the goal. Therefore, complex shaped obstacles with many large concavities do not create local minima of the overall potential even for a complex robot. Local minima can occur only when the workspace is so cluttered that the solution path has to come very close to the obstacles. But in that case, the number of discrete configurations lying in freespace (i.e. which do not intersect the obstacles) around the attractor associated with the local minimum can only be very small. The attractor is therefore filled very fast by the best-first search algorithm.

This approach is only valid for a small number n of DOF, because the number of discrete configurations in any given attractor increases exponentially with the number of DOF, even though it is generally small for $n \leq 4$.

However, it has two advantages:

- It is extremely fast: very tricky paths for a 3 DOF bar in a 2D workspace were obtained

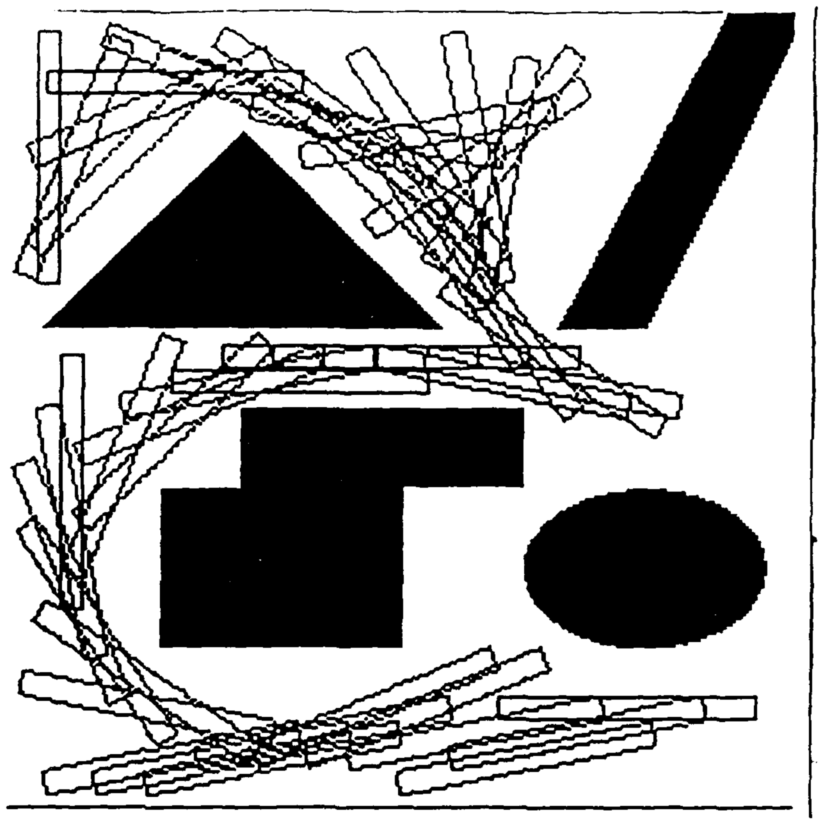

Figure 6: Path generated by a 3 DOF mobile robot in a 2D workspace.

in about 1 second. Simpler paths without complex maneuvers can be obtained in about 1/5 second, which can already be considered real-time.

- It is deterministically resolution complete, i.e. the method is guaranteed to reach the goal in a finite amount of time whenever a solution exists (up to the resolution of the discretization), or return failure when there is no solution.

The next subsection is devoted to some experimental results obtained with this planner.

4.2 Experimental Results

We experimented the path planner on a 'mobile robot' with 3 DOF, namely a long bar in a 2D workspace.

The mobile robot is a planar rectangular object with two DOF of translation and one DOF of rotation (see figure 6). Figure 6 shows an example of path generated by the planner. The displayed path shows the ability of the planner to produce complex maneuvers.

The heuristic potential was computed by considering two points on the robot body, at the two extremities of the bar.

In the setting shown in figure 6, the running time of the algorithm was 1 second. This compares extremely favorably (three orders of magnitude faster) with the running time reported in [Brooks 83] for similar path planning problems. One of these orders of magnitude is obtained through the improvement of the computing hardware. The two other ones are actually a product of our algorithm.

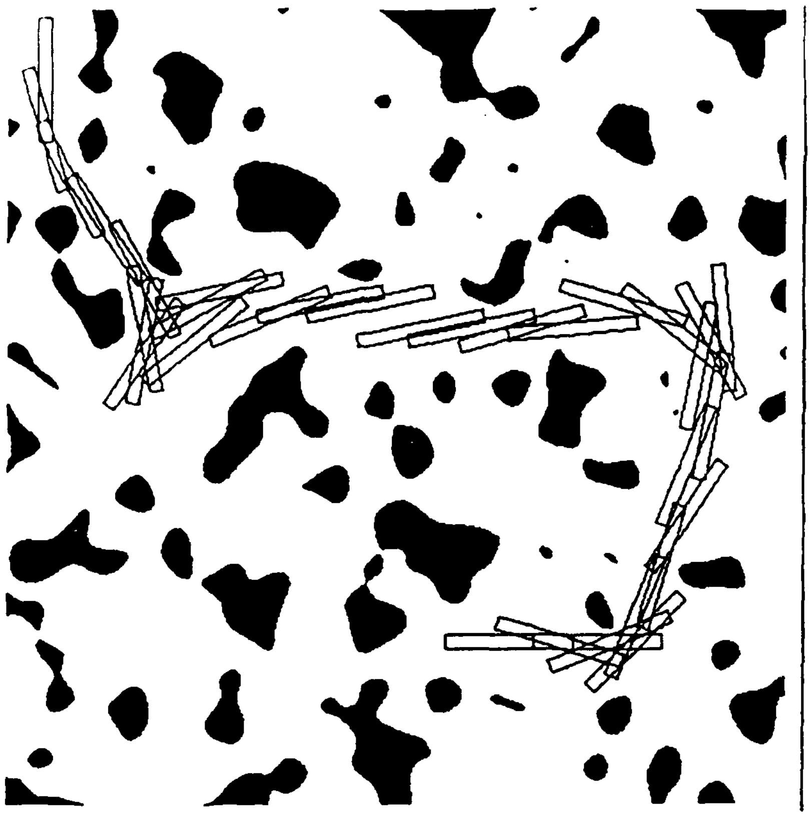

Figure 7: Another path generated among randomly distributed obstacles.

Figure 7 shows another example of the abilities of the algorithm. Here, the same robot found a path in less than 5 seconds within a 512^2 bitmap world obstructed by more than 70 complex-shaped obstacles. This example demonstrates a main advantage of distributed representations over centralized ones: the running time of the algorithm does not depend on the number and shape of the obstacles.

This hierarchical best-first search algorithm is the fastest planner developed so far for robots with a small number of DOF. However, we had to develop another method based on a Monte-Carlo procedure in order to deal with many DOF. Of course, the Monte-Carlo-based algorithm also works for a small number of DOF. We experimented it in the same setting as the one displayed in figure 6. The result obtained (after energy minimization) is shown in figure 8. The running time was 10 second on the same hardware, which is ten times slower. However, as we will see in section 7, the Monte-Carlo based algorithm is massively parallelizable, and will certainly prove extremely faster than the best-first search of dedicated computing hardware. We believe that real-time capabilities are achievable with this algorithm.

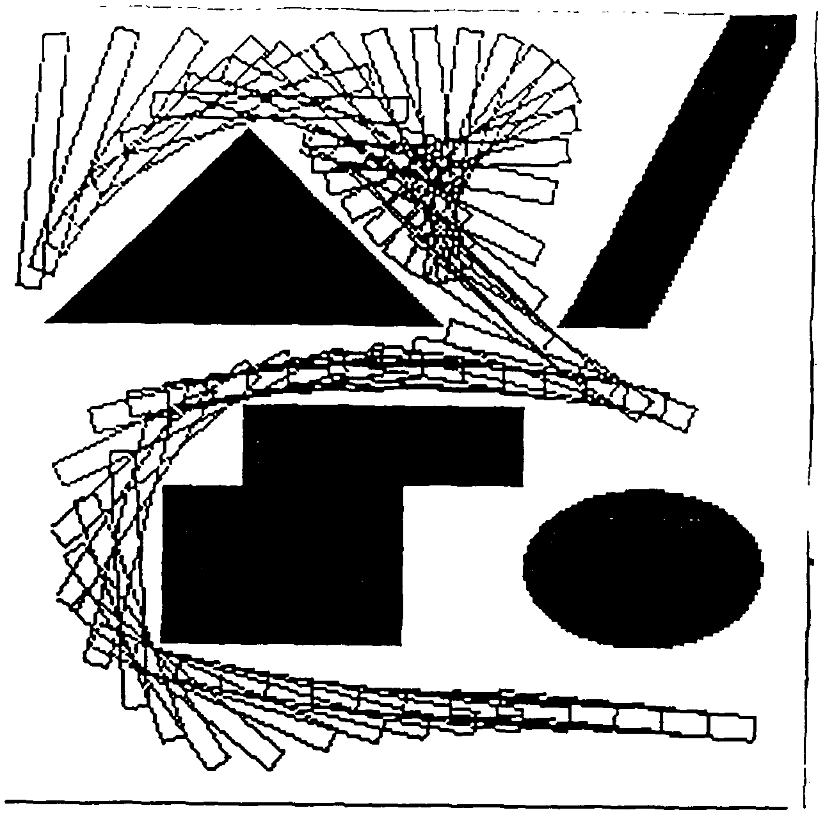

Figure 8: The same path planning problem solved by the Monte-Carlo procedure.

5 The Case of Non-Holonomic Constraints

5.1 Overview

This section introduces the first algorithm able to deal systematically with maneuvers for the 'car problem', i.e. the problem of planning the motion of a car with limited steering angle. This problem was first introduced by [Laumond 86]. Laumond proved that whenever a path in open free space does exist for moving the body of the car without considering any constraint, then a solution to the maneuvering problem exists. However, the solution paths which could be obtained by implementing the constructive proof given in this paper would lead to a very large number of maneuvers, and are therefore not practical. [Laumond 87] proposes an algorithm for planning the motion of the car whenever there is no need to perform maneuvers. The problem of planning maneuvers has been addressed for some particular obstacles configurations by other authors [Tourn 88], but no general solution has been proposed yet. The main idea of our approach is to discretize the problem and perform a dynamic programming search of the solution. The resulting algorithm is again *resolution complete*. Furthermore, optimality under different criteria can be achieved. In particular, we can constraint the algorithm to find the solution path with the minimal number of maneuvers possible.

The next subsection is devoted to a characterization of holonomy for mechanical systems using Frobenius Integrability Theorem. This characterization allows us to prove the non-holonomicity of the 'car problem'. The reader familiar with holonomy may want to skip this subsection.

5.2 A Local Characterization of Linear Holonomy.

We have considered so far the motion planning problem as a pure geometric problem, i.e. a problem for which all the constraints can be expressed using only the geometric variables $(q_1, \dots, q_n)$. In other words, a constraint is said to be geometric when is can be expressed as an equation in configuration space:

$$F(q,t) = 0, \qquad q \in C \tag{1}$$

Such a constraint can be interpreted geometrically in the following way (if the function F is smooth and has a surjective differential): equation 1 defines a submanifold of the configuration space C, and it means that the mechanical system considered is bound to stay on this submanifold whatever force is applied to it.

A real world mechanical system can have many kinds of constraints, some of which can be reduced to a geometrical relation, some others which cannot. In particular, one can consider the larger class of *kinematic constraints*, i.e. the constraints which include the geometric parameters and/or velocity parameters. Mathematically, any kinematic constraint can be expressed as:

$$G(q,\dot{q},t) = 0, \qquad (q,\dot{q}) \in TB(C) \tag{2}$$

where $TB(C)$ is the *tangent bundle* associated with the manifold C, i.e. the space representing the configurations and velocities. The tangent bundle is itself a manifold [Spivak 79]. It is more commonly called *phase space* in Physics, or *state space* in control terminology.

When dealing with a general kinematic constraint of the form 2, an interesting problem is to know if the constraint can be *integrated*, i.e. if the system of first order differential equations defined by 2 can be reduced to a set of geometric relations of the form 1. This problem is commonly referred to in mechanics as the *holonomy problem*.

A constraint of the form 2 will be said to be *holonomic* if it can be integrated, and non-holonomic otherwise. To our knowledge, there is no mathematical result about the integrability of equations of the form 2 in the general case. However, there is a very important particular case for which a characterization can be derived. We detail this particular case in the next few paragraphs.

Kinematic constraints on mechanical systems are generally the result of a *rolling contact* between two rigid bodies. For example, in the case of the 'car problem', the kinematic constraint is the result of the rolling of the wheels on the ground. The kinematic constraint is expressed as a relation between the relative velocities of the two points in contact. In general, the contact is a combination of rolling and sliding, this combination depending on the friction coefficients of the two bodies. When there is no sliding at all, the relation simply states that the velocities of the two points in contact are the same, relation which is linear in the velocity parameters. Therefore, the case where the function G of formula 2 is *linear* as a function of the velocities is particularly interesting. Under the assumption of perfect rolling, equation 2 can be rewritten:

$$G(q,\dot{q},t) = \omega(q,t).\dot{q} = \sum_{i=1}^{i=n} \omega_i(q,t)\dot{q}_i = 0 \tag{3}$$

For each configuration q, $\omega(q,t)$ is linear. By definition, ω is called a *1-differential form* ([Spivak 79]). For each $q \in C$, equation 3 defines an hyperplane $\Delta(q)$ included in the tangent space of C at q. The function Δ is called *(n-1)-distribution* associated to ω.

There is a fundamental result about integrability of *linear kinematic constraints* called *Frobenius Integrability Theorem*, which can be simply stated as follows when applied to 1-forms:

Theorem 1 (Frobenius Integrability Theorem)

Let ω be a 1-form on a manifold C of dimension n, and Δ the associated distribution of hyperplanes.

In a neighborhood of any point $q_0 \in C$, the three following conditions are equivalent:

i) $\omega \wedge d\omega = 0$ *(i.e. the exterior product between ω and its exterior differential is null).*

ii) *The distribution Δ is invariant under the Lie bracket. (i.e. for any couple of vector fields (X,Y) on Δ, $[X,Y]$ is also on Δ).*

iii) *There is a foliation of C tangent to Δ. (i.e. the constraint 3 can be integrated)*

A proof of this theorem in the general case can be found in [Spivak 79]. In the case of 1-forms stated above, a pedestrian proof only based on elementary calculus (Fixed point theorem) can be found in [Barra 88]. It is important to notice that this local result can be globalized ([Spivak 79], vol. 1, pp 264-268). From this theorem, we are able to infer a local characterization of holonomy for linear kinematic constraints, and in particular of any kind of kinematic constraint induced by pure rolling contacts.

By definition of exterior differentiation of differential forms, we have:

$$d\omega = \sum_{1\leq i<j\leq n} \left(\frac{\partial \omega_j}{\partial q_i} - \frac{\partial \omega_i}{\partial q_j}\right) dq_i \wedge dq_j$$

From the definition of exterior product of differential forms and the above formula we compute:

$$\omega \wedge d\omega = \sum_{1\leq i<j<k\leq n} \left(\omega_i \left(\frac{\partial \omega_k}{\partial q_j} - \frac{\partial \omega_j}{\partial q_k}\right) + \omega_j \left(\frac{\partial \omega_i}{\partial q_k} - \frac{\partial \omega_k}{\partial q_i}\right) + \omega_k \left(\frac{\partial \omega_j}{\partial q_i} - \frac{\partial \omega_i}{\partial q_j}\right)\right) dq_i \wedge dq_j \wedge dq_k$$

Therefore, we deduce from Frobenius Theorem:

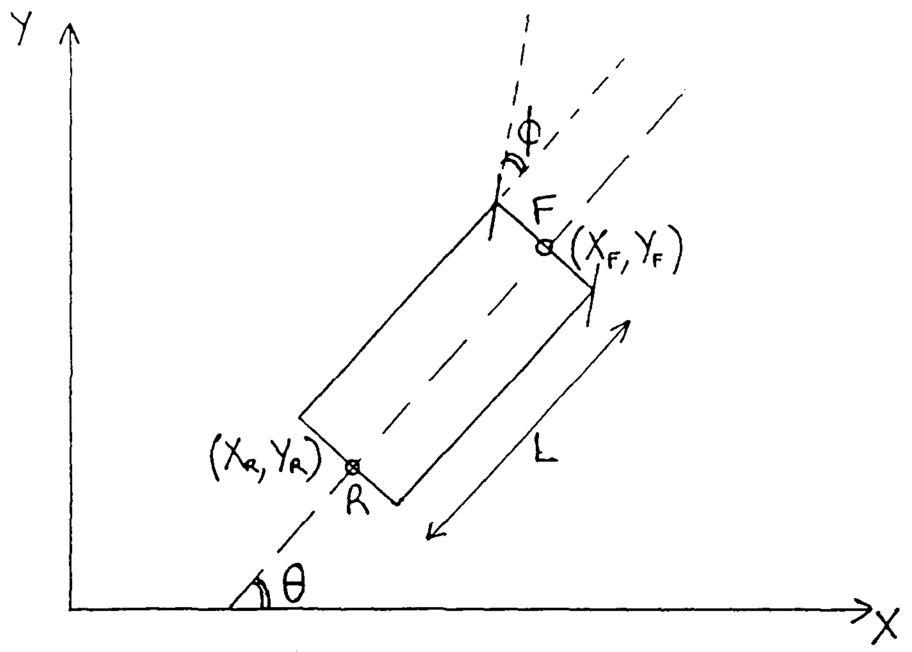

Figure 9: The Car Problem.

Corollary 1 (Characterization of linear holonomy)
A linear kinematic constraint (and in particular a pure rolling constraint) defined by:

$$G(q,\dot{q},t) = \omega(q,t).\dot{q} = \sum_{i=1}^{i=n} \omega_i(q,t)\dot{q}_i = 0$$

is holonomic if and only if the following relation holds for any $(i,j,k) \in [1,n]^3$ such that $1 \leq i < j < k \leq n$:

$$A_{ijk} = \omega_i \left(\frac{\partial \omega_k}{\partial q_j} - \frac{\partial \omega_j}{\partial q_k}\right) + \omega_j \left(\frac{\partial \omega_i}{\partial q_k} - \frac{\partial \omega_k}{\partial q_i}\right) + \omega_k \left(\frac{\partial \omega_j}{\partial q_i} - \frac{\partial \omega_i}{\partial q_j}\right) = 0$$

In the next subsection, we state the 'car problem', and prove its non-holonomicity using the above result. This property implies that the 'car problem' cannot be handled by purely geometric algorithms.

5.3 The Car Problem

Let us consider a front wheel drive car with limited steering angle. Figure 9 displays a simple model of such a car viewed from the top. Such a mechanical system has 3 DOF (two translations and one orientation), but it cannot change the three configuration space parameters independently. This is so because the velocity of the car rear is always tangent to the car orientation: the car has a kinematic constraint[2].

The car configuration is parameterized by the coordinates X_f and Y_f of the driver (the point F in the middle of the two front wheels), and the car orientation θ. The steering angle of the front wheels is denoted by $\phi \in [-\phi_{max}, +\phi_{max}]$. The car can move forward $v \geq 0$ or backward $v \leq 0$. The length of the car is L.

The coordinates (X_r, Y_r) of the point R in the middle of the rear wheels verify the following relation:

$$X_r = X_f - L\cos\theta \qquad Y_r = Y_f - L\sin\theta \tag{4}$$

The kinematic constraint expresses that the velocity of R is parallel to the orientation of the car:

$$\dot{X}_r = \lambda\cos\theta \qquad \dot{Y}_r = \lambda\sin\theta$$

where λ is a constant. Eliminating λ, we get:

$$-\dot{X}_r\sin\theta + \dot{Y}_r\cos\theta = 0 \tag{5}$$

Taking the derivative of formula (4) we get:

$$\dot{X}_r = \dot{X}_f + \dot{\theta}L\sin\theta \qquad \dot{Y}_r = \dot{Y}_f - \dot{\theta}L\cos\theta \tag{6}$$

Combining (5) and (6) we obtain the *kinematic constraint*:

$$-\dot{X}_f\sin\theta + \dot{Y}_f\cos\theta - \dot{\theta}L = 0 \tag{7}$$

Using the notations of the previous subsection, we identify:

$$\omega_{X_f} = -\sin\theta \quad \omega_{Y_f} = \cos\theta \quad \omega_\theta = -L$$

We compute the coefficient $A_{X_fY_f\theta}$:

$$A_{X_fY_f\theta} = \omega_{X_f}\left(\frac{\partial\omega_\theta}{\partial Y_f} - \frac{\partial\omega_{Y_f}}{\partial\theta}\right) + \omega_{Y_f}\left(\frac{\partial\omega_{X_f}}{\partial\theta} - \frac{\partial\omega_\theta}{\partial X_f}\right) + \omega_\theta\left(\frac{\partial\omega_{Y_f}}{\partial X_f} - \frac{\partial\omega_{X_f}}{\partial Y_f}\right) = 1 \neq 0$$

Therefore, *the kinematic constraint of the car problem is non-holonomic.*

We now derive the equations governing the control of the system. As the car is a front wheel drive, the velocity of F is computed as:

$$\dot{X}_f = v\cos(\theta + \phi) \qquad \dot{Y}_f = v\sin(\theta + \phi) \tag{8}$$

[2]Here we model this constraint using only the 3 DOF necessary to plan the motion of the car. In fact, a real car has many more DOF and many kinematic constraints. In particular, the constraint that we are simply modeling here is induced by the pure rolling contacts between the wheels and the ground. We do not need to go into such a detailed modelization for our purpose.

Combining this with the kinematic constraint (7) we get:

$$v \sin \phi - L\dot{\theta} = 0$$

Finally, we obtain the full system of differential equations relating the control parameters (v, ϕ) to the configuration parameters (X_f, Y_f, θ) by combining formula (8) and:

$$\dot{\theta} = v\frac{\sin \phi}{L}$$

This system of differential equations is non-linear, and has to be integrated numerically when the controls are time-varying. However, whenever v and ϕ remain constant, it can be solved analytically. There is nothing surprising with that: with a given constant steering angle and at a given speed, the trajectory of the point F is an arc of circle, while the orientation of the car changes linearly. The integration is straightforward. We first compute θ:

$$\theta(t) = \theta(0) + t\frac{v \sin \phi}{L} \tag{9}$$

and then X_f and Y_f using the above result:

$$X_f(t) = X_f(0) + \frac{L}{\sin \phi}\left(\sin(\phi + \theta(0) + t\frac{v \sin \phi}{L}) - \sin(\phi + \theta(0))\right) \tag{10}$$

$$Y_f(t) = Y_f(0) - \frac{L}{\sin \phi}\left(\cos(\phi + \theta(0) + t\frac{v \sin \phi}{L}) - \cos(\phi + \theta(0))\right) \tag{11}$$

From the two last formulas, we deduce that the *turning radius* of the car is

$$R_T = \frac{L}{\sin \phi}$$

The algorithm we use to find a path between two given configurations is very similar to the one used in the holonomic case. The main difference comes from the computation of the neighbors of a given configuration. In the holonomic case, control space and configuration space are identical. This is not any more true in the non-holonomic case. The idea here is to discretize the *control parameters* (v, ϕ) instead of discretizing the configuration parameters. For example, we can set the number of neighbors to 6, corresponding to the 6 following control values:

$$v \in \{-v_0, v_0\}, \quad \phi \in \{-\phi_{max}, 0, +\phi_{max}\}$$

The configurations of the 6 neighbors are then computed from formulas (9), (10), (11).

We could perform the search in a best-first fashion like we did in the holonomic case. This would result in admissible paths, but these solutions would include a very large number of maneuvers. The objective function used in the dynamic programming search is in general a combination of the heuristic potential field term and the term relative to the number of

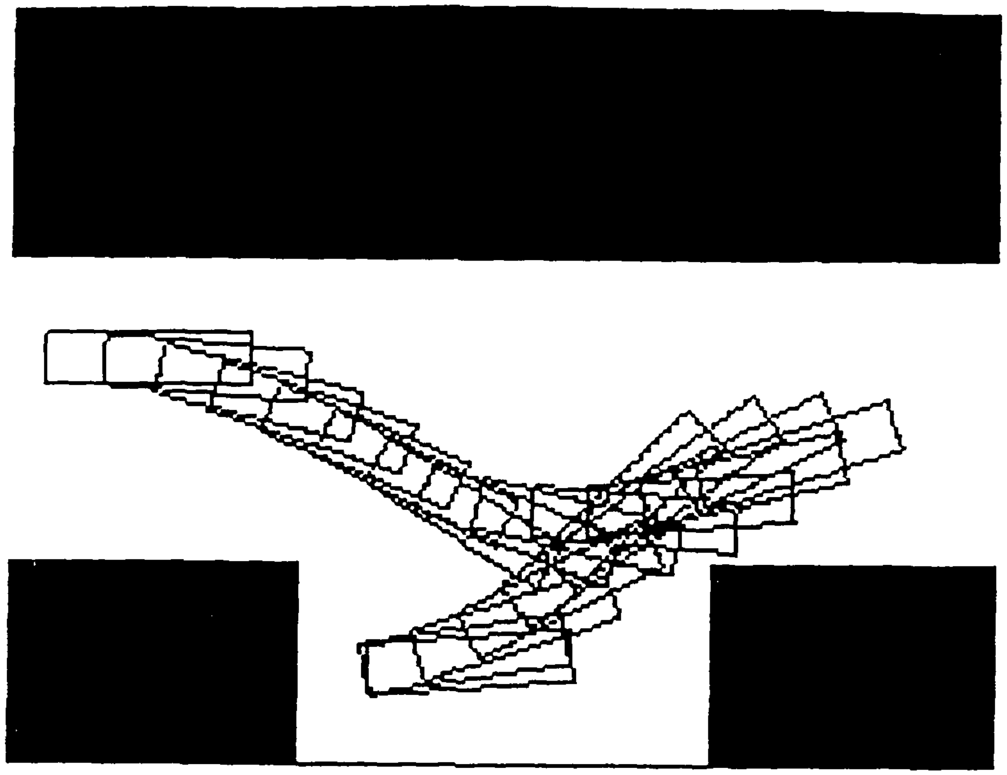

Figure 10: A Parking Maneuver.

maneuvers. If we want to minimize the number of maneuvers, we have to modify the best-first procedure. The idea is to use the precomputed distance to the obstacles D_1, and to compare it to the turning diameter $D_T = 2R_T = 2L/\sin\phi_{max}$. As long as the robot is far from the obstacles and the goal ($D_1 > D_T$), we can proceed in a best-first fashion with no risk to need maneuvers. When the robot comes too close to the obstacles or the goal ($D_1 \leq D_T$), we have to abandon the potential field heuristic and to perform the search using only the term relative to the number of maneuvers.

Examples of paths for the car problem with complex maneuvers are shown and discussed in the next subsection.

5.4 Experimental Results

We experimented the non-holonomic car planner with various values of the maximal steering angle ϕ_{max}. Figure 10 shows an example of the parking problem with a very limited steering angle $\phi_{max} = 30$ degrees. The running time was 30 seconds. This is much longer than the running time in the holonomic case. The reason is that with such a limited steering angle. the turning diameter is larger than the workspace diameter itself. Therefore, the heuristic potential field does not help much in that case to obtain the final solution. The algorithm has to explore a large part of the search space.

Figure 11 shows an example of maneuvering in a clustered environment with a steering angle $\phi_{max} = 45$ degrees. The running time was about 1 minute. Figure 12 displays four

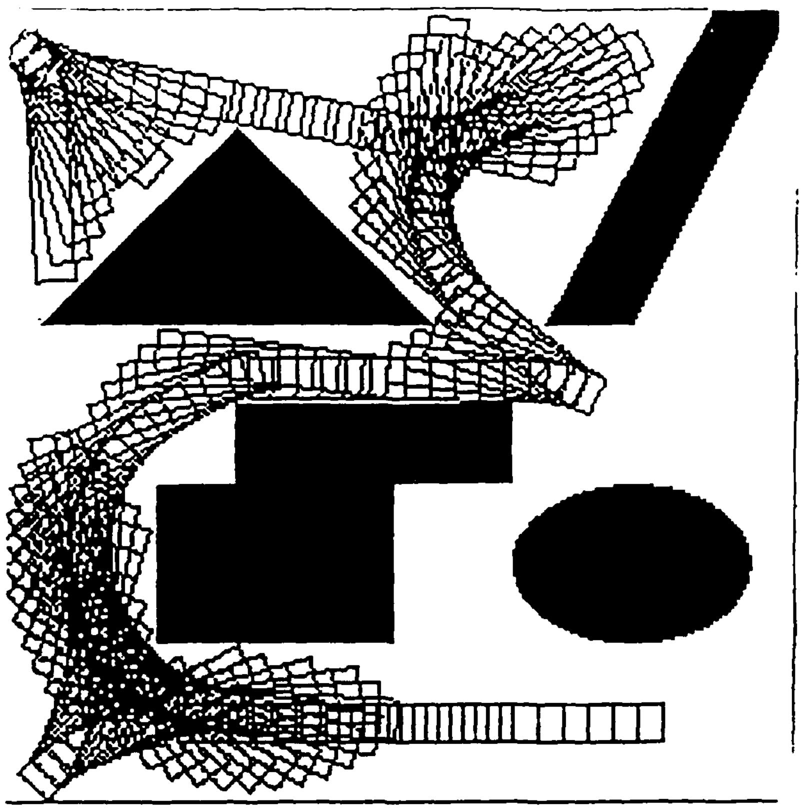

Figure 11: Maneuvering in a clustered environment.

successive parts of the same path so that the different maneuvers can be isolated from each other. Ten maneuvers (i.e. changes on the sign of v) were necessary in this example.

Any kind of maneuver can be performed in any obstacle distribution, up to the resolution of the bitmap.

6 Large Systems: A Monte-Carlo Approach

6.1 Overview

The hierarchical best-first search algorithm described in the previous sections cannot plan the motion of robots with many DOF. The reason of this limitation is the same for any complete motion planning algorithm based on the configuration space approach: the volume of configuration space, which is locally diffeomorphic to $\mathbf{R}^n$, increases exponentially with n. One the other hand, human beings are able to solve motion planning problems with a very high number of DOF. Often, the redundancy of DOF is helpful and simplifies the planning problem. There is a divorce between the mathematical result of NP-hardness and the everyday life experience. To design realistic efficient motion planners for many DOF, we *have to* drop the completeness requirement. Then, the main question is the following: Is it possible to design non-complete algorithms which give nevertheless the correct solution in most real-life cases?

The main purpose of this section is to show some experimental results which let us think

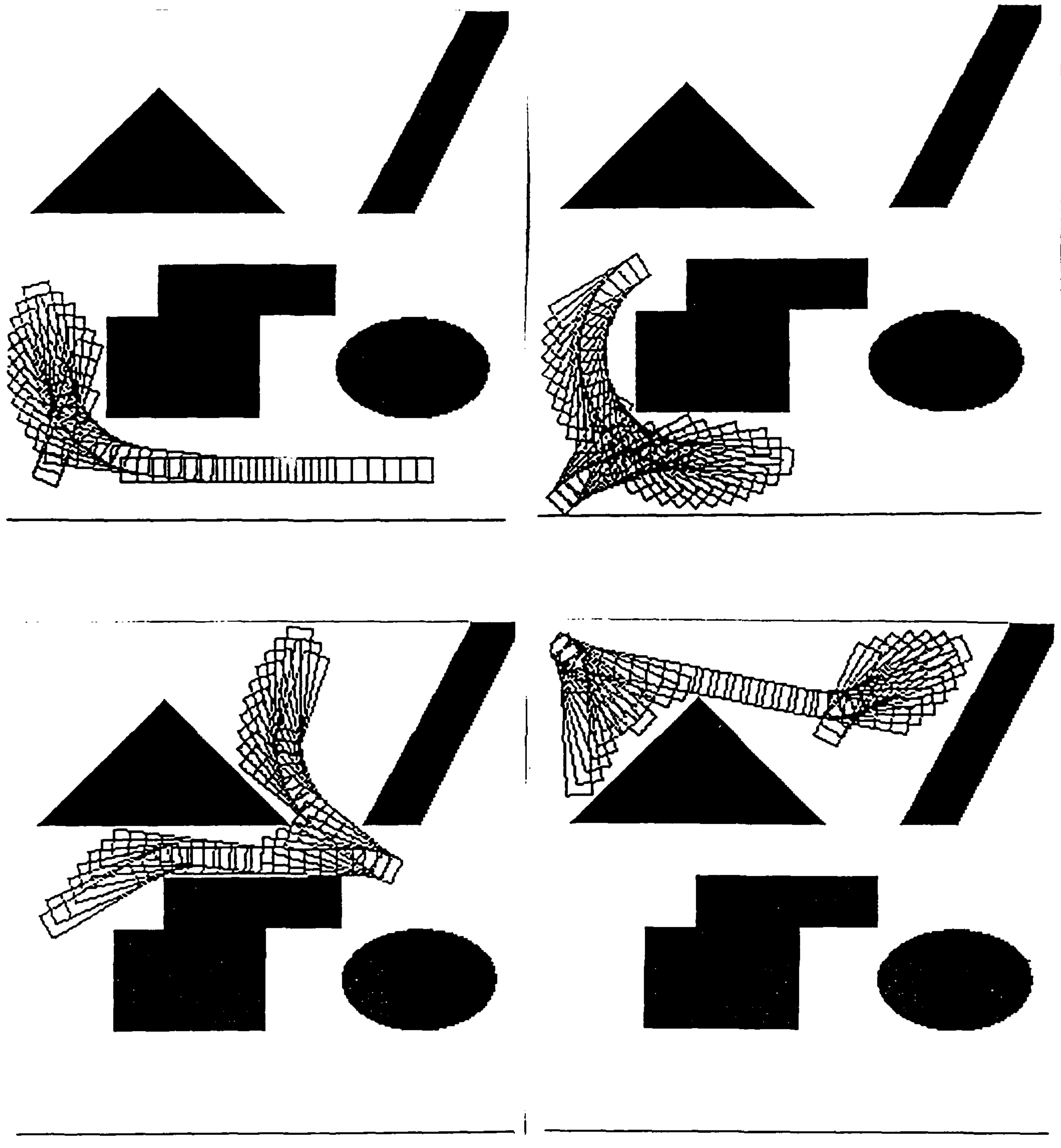

Figure 12: Maneuvering in a clustered environment (continued).

that the answer is positive.

6.2 Local Minima: The Exit Problem

The Monte-Carlo approach that we describe in this subsection uses the same framework as the graph search approach. It uses the same numerical potential field U, and is implemented with the same coarse-to-fine hierarchical decomposition of the world model. As there is no interaction between the search at different levels of hierarchy, all these levels could be implemented concurrently on a parallel machine. The difference comes from the way local minima of the potential field are treated. We detail here the procedure employed at a given resolution level. The principle is the following.

Starting from the initial configuration q_{INIT}, we first follow the gradient of the numerical potential field $U(q)$ until we reach a local minimum q_{loc} at altitude $U_{loc} = U(q_{loc})$. The problem of exiting from this local minimum is solved by generating several random motions from q_{loc}, each of these random motions having a random duration. The two key points of the algorithm are how to perform these random motions, and how to choose their duration. Before detailing these two points in the next subsection, we give the overall sketch of the algorithm. From all the termination configurations of these random motions, we follow again the gradient of the potential field until we reach new local minima. We retain only those of the new local minima whose altitude is lower than U_{loc}. If none of the new local minima has an altitude lower than U_{loc}, we say that q_{loc} *is a dead-end.* Otherwise, for each of those new local minima retained, we perform again the random motion procedure, in a best-first fashion. So, a graph of local minima is incrementally built, the path joining two neighbor local minima being the concatenation of a random motion and a gradient motion. A standard best-first search of this graph is performed until we reach the goal, or until all the local minima developed are dead-ends.

When the solution path is finally obtained, we smooth it using the classical principles of Variational Calculus. More precisely, we minimize the total work needed to move the robot from the initial configuration to the goal configuration.

A very nice property of this procedure is that all the brownian motions starting from a given local minimum can be performed in parallel on any SIMD machine (there is no need for intercommunication between the different processing units).

As we use a random procedure to build the graph of local minima, we can never be sure to find a path whenever it exists. In other words, the technique is not deterministically resolution-complete. However, the fundamental properties of Brownian motion allow to prove that when the computation time converges towards infinity, the probability to reach the goal converges towards 1. We will say that our Monte-Carlo procedure is *probabilistically resolution-complete.* This convergence-in-distribution property, well-known for the so-called 'simulated annealing' algorithms [Geman 86], is a very weak one. To illustrate its weakness, let us consider the most silly (uninformed) of the motion planning algorithms that one can imagine: Start a Brownian motion from the initial configuration without using any heuristic, and keep on until reaching a small neighborhood of the goal configuration. This entirely uninformed algorithm is probabilistically resolution-complete! What the theory does not tell is how long time the algorithm will need to actually reach the goal (and it should better not tell!). Designing algorithms only makes sense when one has some control of the computational complexity in the average case.

The potential-field-informed Monte-Carlo procedure that we propose happens to be experimentally efficient for motion planning. Again, the efficiency of such a simple technique may appear fairly surprising. We believe that the reason of its efficiency is the following: The mathematical problem of motion planning when stated in algebraic terms is known to be Pspace-hard in the number n of DOF. On the other hand, the more DOF a robot manipulator has, the more paths there are to reach a given configuration from any initial configuration. The redundancy of DOF often appears helpful when human beings are confronted to a given planning problem. In other terms, the more DOF you have, the more solutions to the planning problem you have. The worst-case behaviour of the problem (very large search space with very few solutions) does not appear often in real-life cases. The average case behaviour seems to be more of the type: very large search space with very large number of solutions. In such cases, controlled stochastic algorithms can be particularly well suited.

Monte-Carlo procedures have already been helpful in some cases for solving NP-hard problems. [Cerny 85] gives sub-optimal solutions to the traveling salesman problem for more than 10000 towns. [Kirk 83] proposes an algorithm better than human experts for placement and routing of VLSI chips. The traveling salesman problem is indeed NP-hard, but the very large search space is associated with a very large number of 'good' sub-optimal solutions. It is the same for routing of VLSI.

Monte-Carlo procedures for optimization have also been used more or less successfully in many other fields of science and engineering, and in particular in Image Processing. [Geman 84] applies a so-called 'simulated annealing' approach for restoring images blurred with non-linear filters. Several authors have implemented edge detection algorithms based on the same paradigm. Recently, it has been applied to higher level problems in Computer Vision. [Barnard 88] addresses the stereo matching problem using a hierarchical pixel-level simulated annealing algorithm. [Barra 88] proposes a new approach to the 3D reconstruction of stratigraphic layers and the detection of geological faults in seismic data. Despite some impressive results for specific problems, the Monte-Carlo approach to Computer Vision cannot be considered as a general paradigm. There is a straightforward reason to this lack of popularity: the search space of Computer Vision problems is huge. Furthermore, these problems are hard to express correctly in terms of the optimization of a functional. When it is possible, several parameters have to be tuned empirically or by means of identification models, leading to even heavier optimization problems.

For motion planning, the number of parameters has been reduced to a tractable set using the specific characteristics of the problem, and the function to optimize is well defined. Furthermore, the Monte-Carlo procedure proposed behaves very differently from the classical 'simulated annealing' procedures. Basically, simulated annealing procedures perform a kind of breadth-first search of the graph of local minima of the function to be optimized, whereas our Monte-Carlo algorithm performs a best-first search of this graph using as heuristic the function to optimize itself. We believe that this best-first approach leads to much higher convergence rates. This belief is supported by strong experimental evidence in the case of Robot Motion Planning.

6.3 What Is a Continous Random Motion?

We now detail the first key point of the algorithm, that is how to perform a random motion. When we reach a local minimum of the potential field U, there is no more information that we can extract locally from U to guess the direction of motion which will lead to the goal. In some way, the potential gave us all what it knows locally about the goal. ***As long as we do not make any assumption on the statistics of the obstacles distribution,*** we have no more piece of valuable information to reach the goal. Then, if we do not want our algorithm to stop and return failure, we have to choose the next direction without any information. As we work at a given resolution $R_i = -\log_2 \Delta_i$ for each configuration space parameter q_i, we would like the standard deviation of our step to be equal to Δ_i. But the most uninformed distribution (i.e. the one which maximizes the entropy) among all those which have a given standard deviation Δ_i is precisely the Gaussian distribution [Roubine 70]. Therefore, our velocity along each configuration parameter q_i must follow a Gaussian distribution with deviation Δ_i. But the problem of choosing our velocity happens again and again at every single time, independently of the previous times. Therefore, the velocity process $n(t)$ must be a time-decorrelated Gaussian process. By time-decorrelated we mean that the autocorrelation function is:

$$\Gamma_n(t,t') = \Delta_i^2 \delta(t-t')$$

where δ is the Dirac function, or equivalently[3] that the power spectrum is flat:

$$S_n(\xi) = \Delta_i^2$$

This is precisely the definition of a *white Gaussian noise* . It can be proven that the white Gaussian noise is the generalized derivative[4] of the *standard Brownian motion*, also called *Wiener-Levy process*. Therefore, the most uninformed motion $Q(t) = (Q_1(t), \ldots, Q_n(t))$ in configuration space is the Brownian motion. The Wiener-Levy process is a non-stationary homogeneous centered Gaussian Markovian process with the following properties [Papoulis 65]:

- Its density is given by:

$$p_i(q_i,t) = \frac{1}{\Delta_i\sqrt{2\pi t}} \exp(-\frac{q_i^2}{2\Delta_i^2 t})$$

[3]This equivalence results from the Bochner-Wiener-Khintchine theorem [Feller 66]: The spectral power is the Fourier transform of the autocorrelation function for any stationary process.

[4]derivative in the sense of stochastic distributions [Roubine 70] [Fernique 67].

- The standard deviation of increments $\Sigma(t)$ increases proportionally to the square root of the duration:
$$\Sigma^2(t) = E\left((Q_i(t_0 + t) - Q_i(t_0))^2\right) = \Delta_i^2 t$$
- The increments are independent, i.e. the two processes $Q(t) - Q(t')$ and $Q(t') - Q(t'')$ are independent for any (t, t', t'')

It can be shown that the Wiener-Levy process is almost surely continous[5], but this is a highly non-trivial result! The Wiener-Levy process is well-defined as long as it does not encounter any obstacles in configuration space. When the process $Q(t)$ hits the boundary of a C-obstacle, the Brownian motion procedure has to be adapted in order to remain in freespace. The classical generalization of Brownian motion when the space is bounded is called ***reflective boundary*** motion. Basically, it consists of reflecting symmetrically to the tangent hyperplane of the C-obstacle the motion that would take place if there were no obstacle. The mathematical consistency of this reflective boundary assumption is fully discussed in [Anderson 76]. We will not detail it in this paper.

We now detail the second key point of the algorithm, that is how to choose the duration of the Brownian motions $Q(t)$. Basically, we want these motions to take the robot out of the local minima of U. For each local minimum q_{loc} of U, we can consider its scope, or *attraction radius* $A_{Ri}(q_{loc})$ in each direction q_i. This scope is the distance, along each configuration space parameter q_i, between q_{loc} and the nearest saddle point q_{saddle}. To escape from a local minimum, we simply have to exit the attractor of q_{loc}. To do so, the minimum distance that we have to travel in each direction q_i from $q_{loc i}$ is precisely $A_{Ri}(q_{loc})$. If we are able to estimate the statistics of A_R, the property $\Sigma(t) = \Delta_i\sqrt{t}$ gives us a clue for the exit problem. The duration $T(q_{loc})$ of the Brownian motions should be of the order of

$$T(q_{loc}) \approx \max_{i\in[1,n]} \left(\frac{A_{Ri}(q_{loc})}{\Delta_i}\right)^2 \tag{12}$$

But to estimate the statistics of A_R, we need to have a statistical model of the potential field U. As we do not make any assumption on the obstacles distribution, we cannot infer any statistical property about U and A_R. Generally, we may assume that the distance A_{Ri} for each parameter q_i will not exceed the value that would provoke a motion in workspace of more than the workspace diameter itself. This diameter being 1 with our normalized coordinates, we obtain the following estimate of A_{Ri} for any local minimum q_{loc}:

$$A_{Ri}(q_{loc}) \approx 1/J^i_{\text{sup}}$$

On the other hand, we have $\Delta_i \approx \delta/J^i_{\text{sup}}$, where δ is the distance between two points in workspace. Combining these two formulas with (12), we obtain:

$$T(q_{loc}) \approx \frac{1}{\delta^2}$$

[5]in other words, the probability for being discontinuous at any given point is zero

We now have to choose for T a single-parameter family of distributions to define completely our algorithm. All what we know about the attraction radii A_{R_i} is that they are strictly positive, and that their expected value is bounded by $1/J^i_{\text{sup}}$. Again, the most uninformed distribution (i.e. the one which maximizes the entropy) of a real positive process whose expected value is given is the ***truncated Laplace distribution.*** The density of A_{R_i} is therefore given by:

$$p(A_{Ri}) = J^i_{\text{sup}} \exp(-J^i_{\text{sup}} A_{Ri})$$

The intuitive interpretation of this result is the following: as no hypothesis is made about the shape of the potential field U, the maximum entropy paradigm states that the rate of variation of the number N of attractors having a given radius A_R is constant. In other words, $1/N \times dN/dA_R$ does not depend on A_R. This means that the overall statistical shape of the potential does not change when the resolution varies. But $1/N \times dN/dA_R$ equals $1/p \times dp/dA_R$. Therefore p varies exponentially with A_R, which precisely defines a Laplace distribution.

Consequently, the relation (12) induces the following distribution for the duration T:

$$p(t) = \frac{\delta}{2\sqrt{t}} \exp(-\delta\sqrt{t}) \tag{13}$$

One can verify that the expected value of this distribution is indeed $1/\delta^2$.

The Monte-Carlo procedure is now completely defined. However, the final path obtain is the concatenation of several gradient motions and Brownian motions. It does not have good dynamic properties. Therefore, we postprocess this path by means of an energy minimization procedure that we describe now.

6.4 Path Optimization

There are several Riemanian metrics that we can consider on the configuration space manifold $\mathcal{C}$. The most natural one is the *kinetic energy* metric, but it becomes a complex expression for large systems. In our implementation, we chose a very simple metric, such that the geodesics of the metric are easy to compute. For some simple unit mass mobile robots (like the long bar studied in section 4.2), this metric is exactly the kinetic energy metric. For long manipulator arms including several revolute joints in serie (like the 10 DOF robot shown in section 6.5), it is no longer the case. We used the metric:

$$ds^2 = \sum_{i=1}^{n} dq_i^2$$

where the q_i are normalized coordinates (i.e. all ranging between 0 and 1). The geodesics equation reduces in this case to:

$$\ddot{\gamma}(t) = 0$$

with

$$\gamma(t_{INIT}) = q_{INIT}, \qquad \gamma(t_{GOAL}) = q_{GOAL}$$

This is a classical quadratic two points boundary value problem under the non-linear obstacle avoidance constraint. It can be solved using standard projected gradient procedures such as Gauss-seidel with projection (on a sequential computer) or Jacobi with projection (on a parallel machine). The final path obtained is smooth, and of course homotopic[6] to the initial path obtained from the Monte-Carlo procedure.

At this point, it is necessary to make some remarks about the topological properties of the configuration space, depending on the dimension of the workspace. If the robot is a solid object moving in a 2D workspace (like the long bar), its configuration space is $\mathbf{R}^2 \times S^1$: $\mathbf{R}^2$ for translations, S^1 for rotations. The Poincaré group[7] of this space is:

$$\Pi(\mathbf{R}^2 \times S^1) \equiv Z$$

This simply means that the robot can make any number of turns, either clockwise or counter-clockwise. Any couple of paths having the same total number of turns are homotopic, but two paths with different numbers of turns can never be homotopic. This result has an important consequence for our energy minimization procedure: if the Monte-Carlo algorithm outputs a path having more turns than necessary, the smoothing procedure with not solve the problem. In fact, some cosmetic improvements can be made to the Brownian motion generator in order to avoid this problem. We will not detail it is this paper.

Now, if the robot is a solid object in 3D (free-flying robot), its configuration space is $\mathbf{R}^3 \times SO(3)$: $\mathbf{R}^3$ for translations, $SO(3)$ (special orthogonal group) for rotations. The Poincaré group of this space is

$$\Pi(\mathbf{R}^3 \times SO(3)) \equiv \{0, 1\}$$

This means that there are only two different homotopy classes: the class of paths which make a total rotation of 2π, and the class of paths which make a total rotation of 0. Any path can be shown to belong to one of these two kinds by changing continuously the axis of rotation. Therefore, even if the Monte-Carlo procedure outputs a path with more turns than necessary, the energy minimization procedure will reduce the number of turns to the optimal possible.

Finally, if the robot is a fixed-base manipulator arm with mechanical stops on the revolute joints, the configuration space is simply connected, and equivalent to an n-cube in $\mathbf{R}^n$. Its Poincaré group is extremely simple:

$$\Pi(\mathbf{R}^n) \equiv \{0\}$$

The homotopy problem does not exist for such manipulators.

[6]i.e. the final path is obtained from the initial one through a *continuous* transformation.

[7]The Poincaré group, also called fundamental group, is the quotient of the set of closed paths by the homotopy equivalence relation. Each element of this group defines in some way the number of turns that a path makes

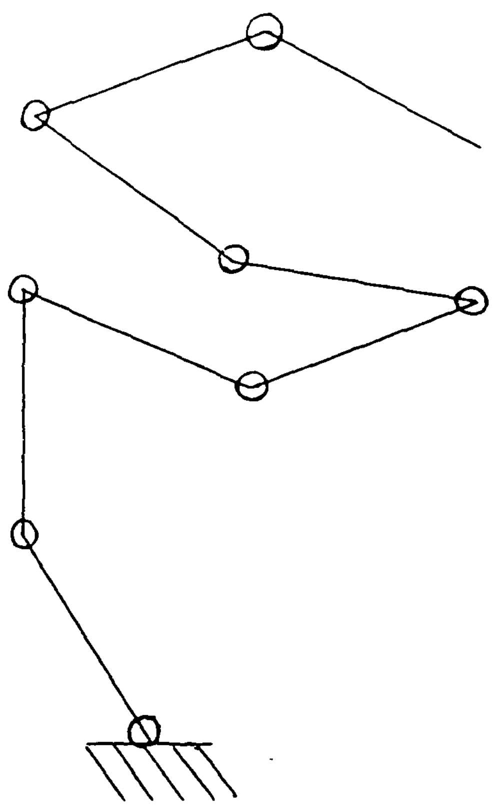

Figure 13: Structure of the 8 DOF serial manipulator.

6.5 Experimental Results

We have tested successfully the Monte-Carlo procedure on several different robots. First, we applied it on the 3 DOF bar as shown in figure 8. The computation time was about ten times slower than with the classical best-first search approach, but we estimate that parallel implementations of the procedure will reduce dramatically the computation time.

Then, we applied the same procedure on a 8 DOF serial manipulator with only revolute joints. The overall structure of this arm is presented in figure 13.

Figures 14 and 15 illustrate two different motions of the robot in a 2D workspace. The paths were generated by the planner.

In the examples shown in figures 14 and 15, we used a heuristic potential U computed with only one point located at the end-effector of the kinematic chain. The algorithm was able to find the solution in 2 minutes for the first example (forward motion), and 30 seconds for the second one (backward motion).

We also applied the procedure on a 10 DOF non-serial manipulator arm including both prismatic and revolute joints. The overall structure of this arm is presented in figure 16.

The 10 DOF manipulator robot includes 7 revolute joints and 3 prismatic joints. For this manipulator, the total number of discrete configurations is of the order of $100^{10} = 10^{20}$ configurations.

Figures 17, and 18 illustrate two different motions of the robot in a 2D workspace. The paths were generated by the planner.

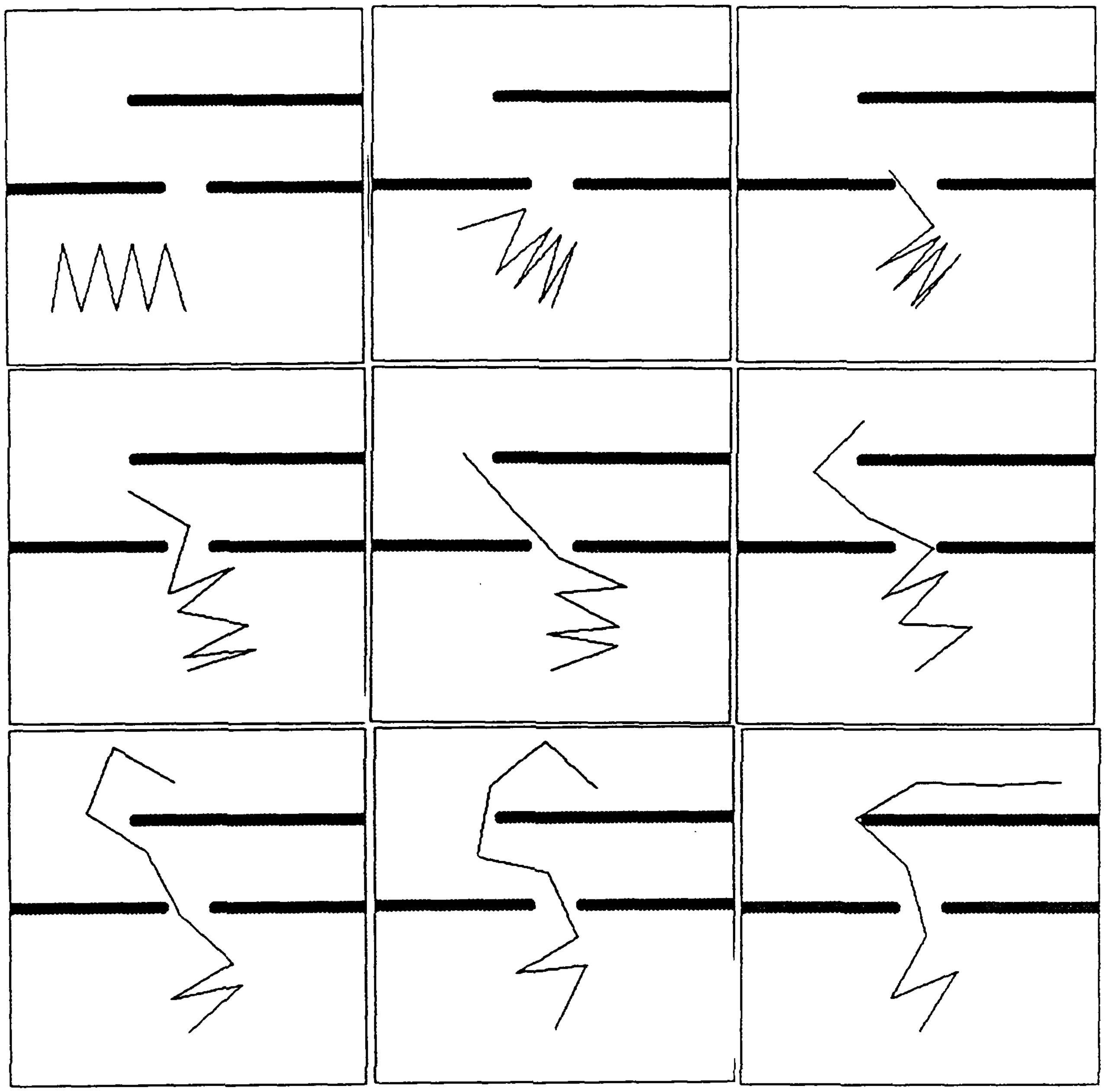

Figure 14: Successive configurations during a forward motion (8 DOF robot).

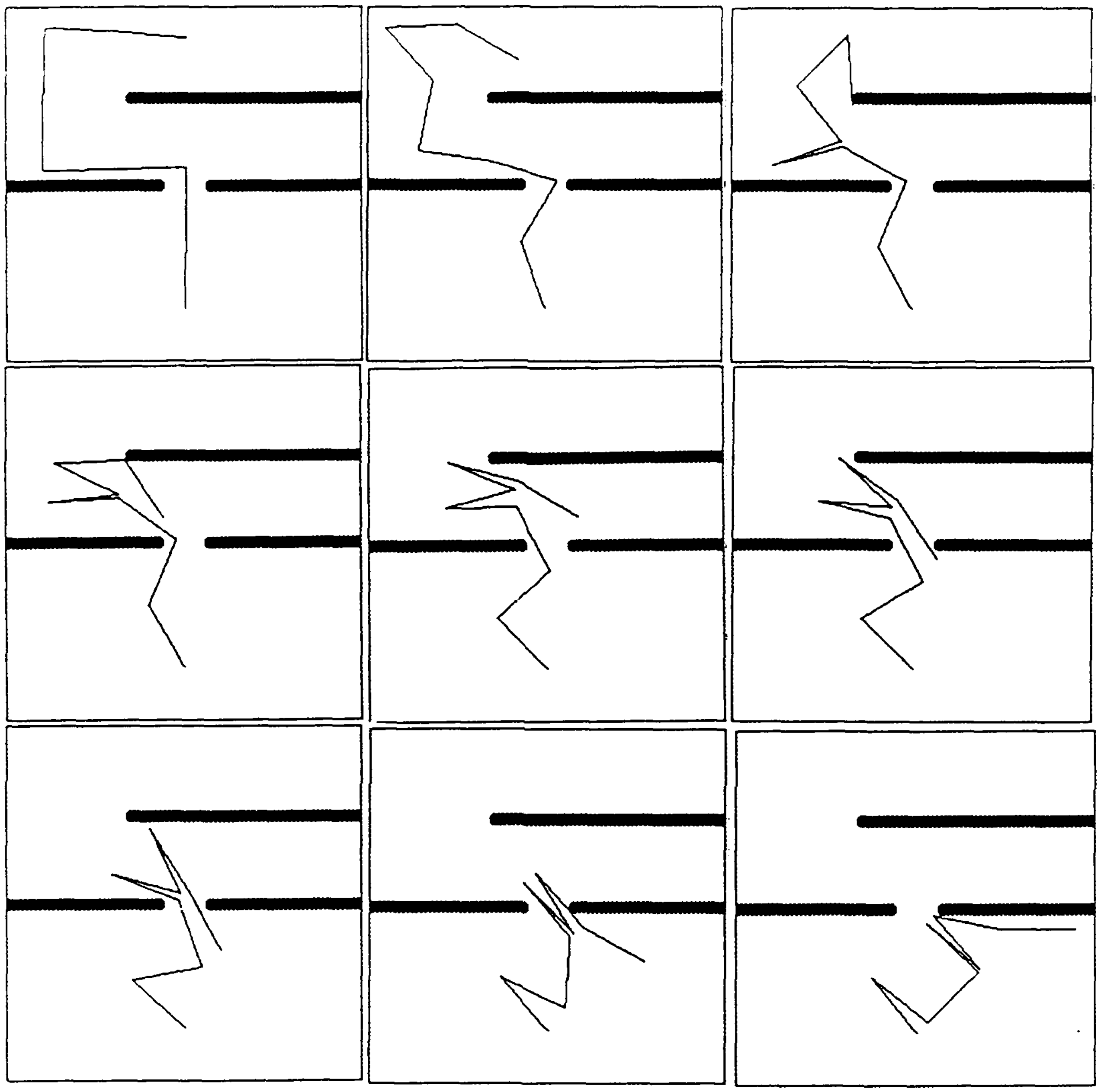

Figure 15: Successive configurations during a backward motion (8 DOF robot).

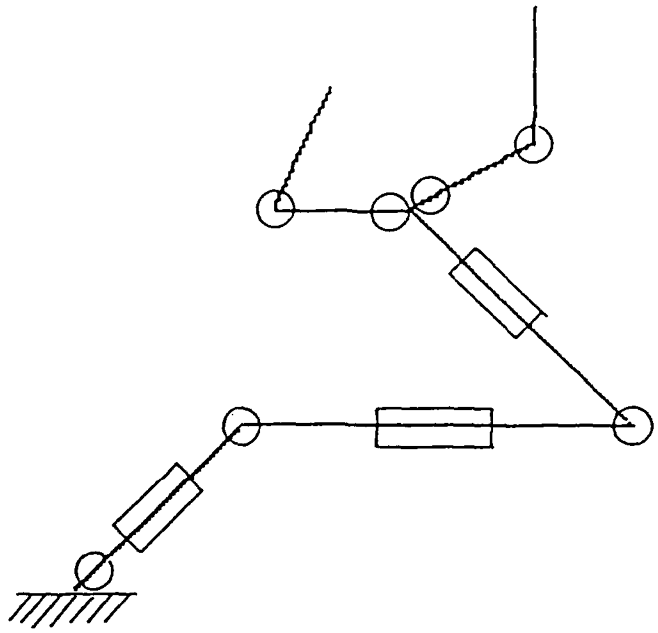

Figure 16: Structure of the 10 DOF manipulator robot.

In the examples shown in figures 17 and 18, we used a heuristic potential U computed with only two points located at the end points of the two kinematic chains. The algorithm was able to find the solution in 3 minutes for the first example, and 2 minutes for the second one.

Finally, we applied the same procedure to a simple example of multi-robot cooperation. Figure 19 displays two 3 DOF mobile robots at the two extremities of a corridor. Figure 20 displays the path found when the goal of the small robot is the upper right corner while the goal of the long robot is the upper left corner. Finally, figure 21 displays the path found when the goal of the two robots is simply to exchange their respective locations. These last examples where solved in about 30 seconds of computation time.

All the examples presented to the planner have been solved within a few minutes of computation. So far, we did not find any case where the planner fails.

The next subsection discusses the completeness issue using the fundamental properties of the Wiener-Levy process.

6.6 Probabilistic Completeness of Monte-Carlo Procedures

In this subsection, we discuss the completeness of the Monte-Carlo procedure described above. Of course, procedures involving random processes can never be deterministically complete. However, weaker completeness results can be obtained under particular circumstances. The algorithm above takes as input a map of the workspace, and a description of

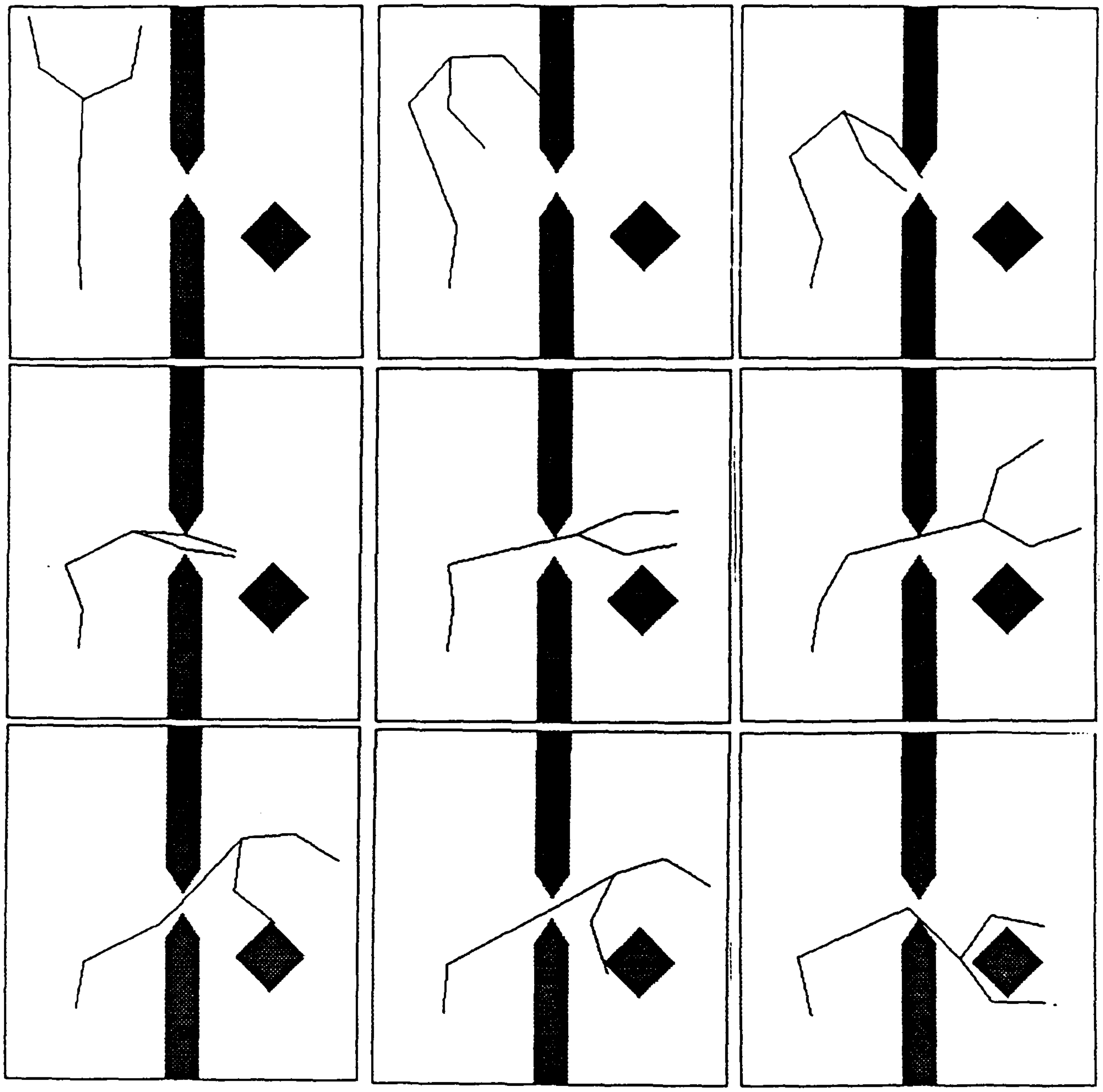

Figure 17: Successive configurations during a forward motion (10 DOF robot).

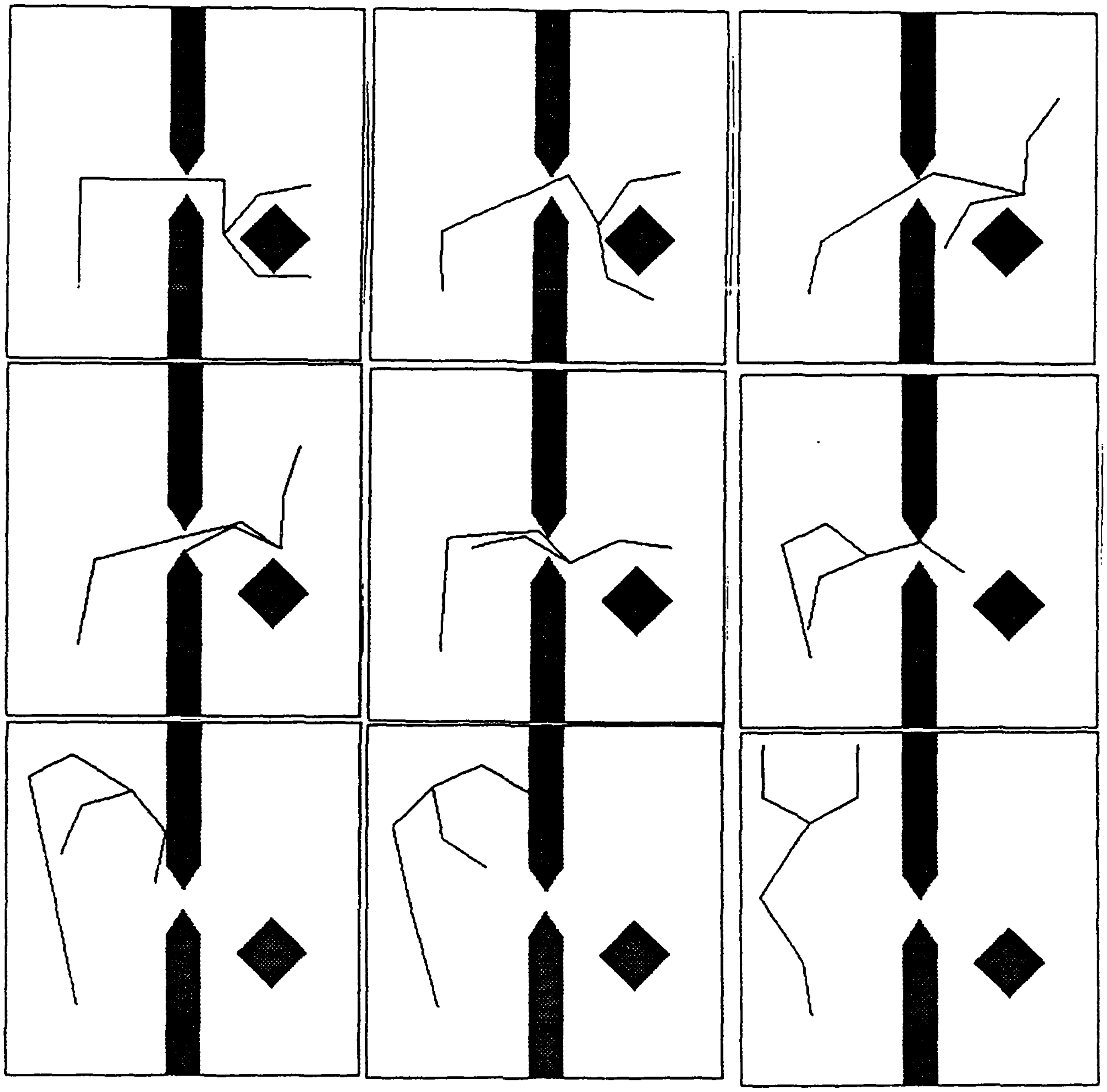

Figure 18: Successive configurations during a backward motion (10 DOF robot).

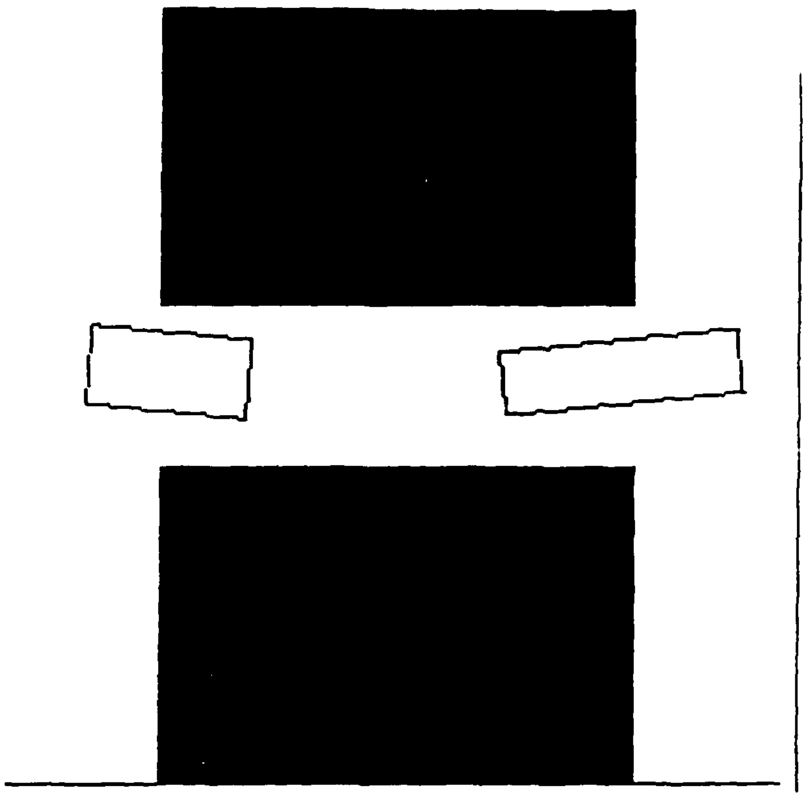

Figure 19: The corridor problem for a 6 DOF multirobot system.

the initial and goal configurations. Its output is two-fold. First the algorithm gives an answer to the existential question: is there a path between the initial and goal configurations? Second, if the path does exist, it provides an explicit solution in configuration space. This may appear surprising: a Monte-Carlo procedure can never give a definitive answer to such an existential question in a finite amount of time! This is why we arbitrarily bounded the number of random motions generated from a given local minimum: the algorithm does not guarantee to answer properly to the existential question, but guarantees to answer it in a finite amount of time.

We claim that if the number of random motions performed from a given local minimum is not bounded, the algorithm weakly converges towards the goal. This claim is based on a very general property of the Wiener-Levy process that we describe now. Whenever the freespace is connected and relatively compact, and given a Brownian motion $w(t)$ with reflective boundaries starting at any point q_{INIT} in freespace, the probability for w to reach during $[0,t]$ any given neighborhood B of the goal q_{GOAL} at least once, converges towards 1 when the computation time converges towards infinity. Formally, we have:

$$\lim_{t \longrightarrow \infty} P(M_B(t)) = 1 \qquad (14)$$

where

$$M_B(t) = \{\exists \tau \in [0,t] \mid w(\tau) \in B\}$$

If statement (14) is true, then our Monte-Carlo procedure obviously converges weakly towards the global minimum of the potential field U. As a matter of fact, we may consider a

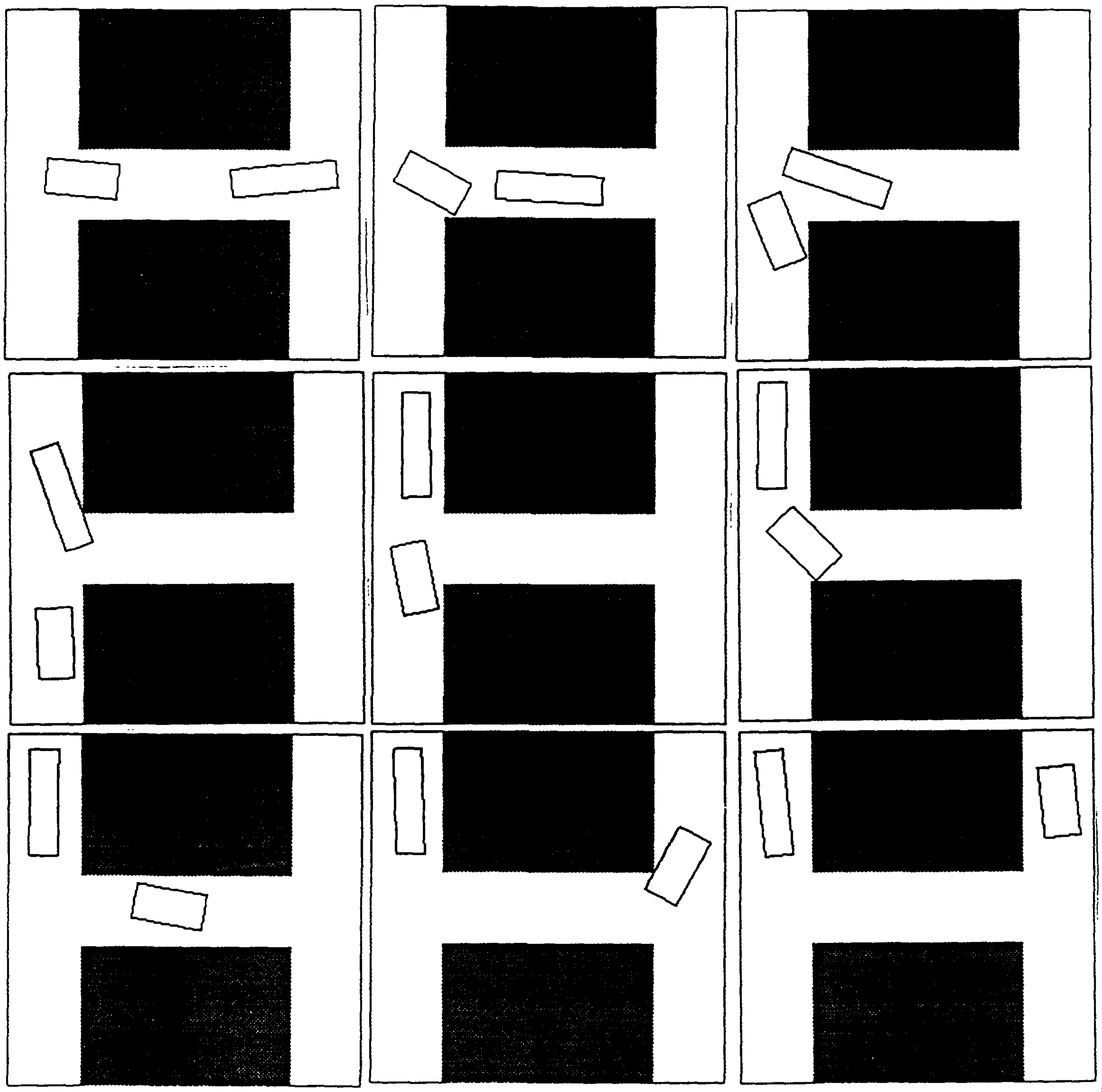

Figure 20: Successive configurations during a simple motion (6 DOF multirobot).

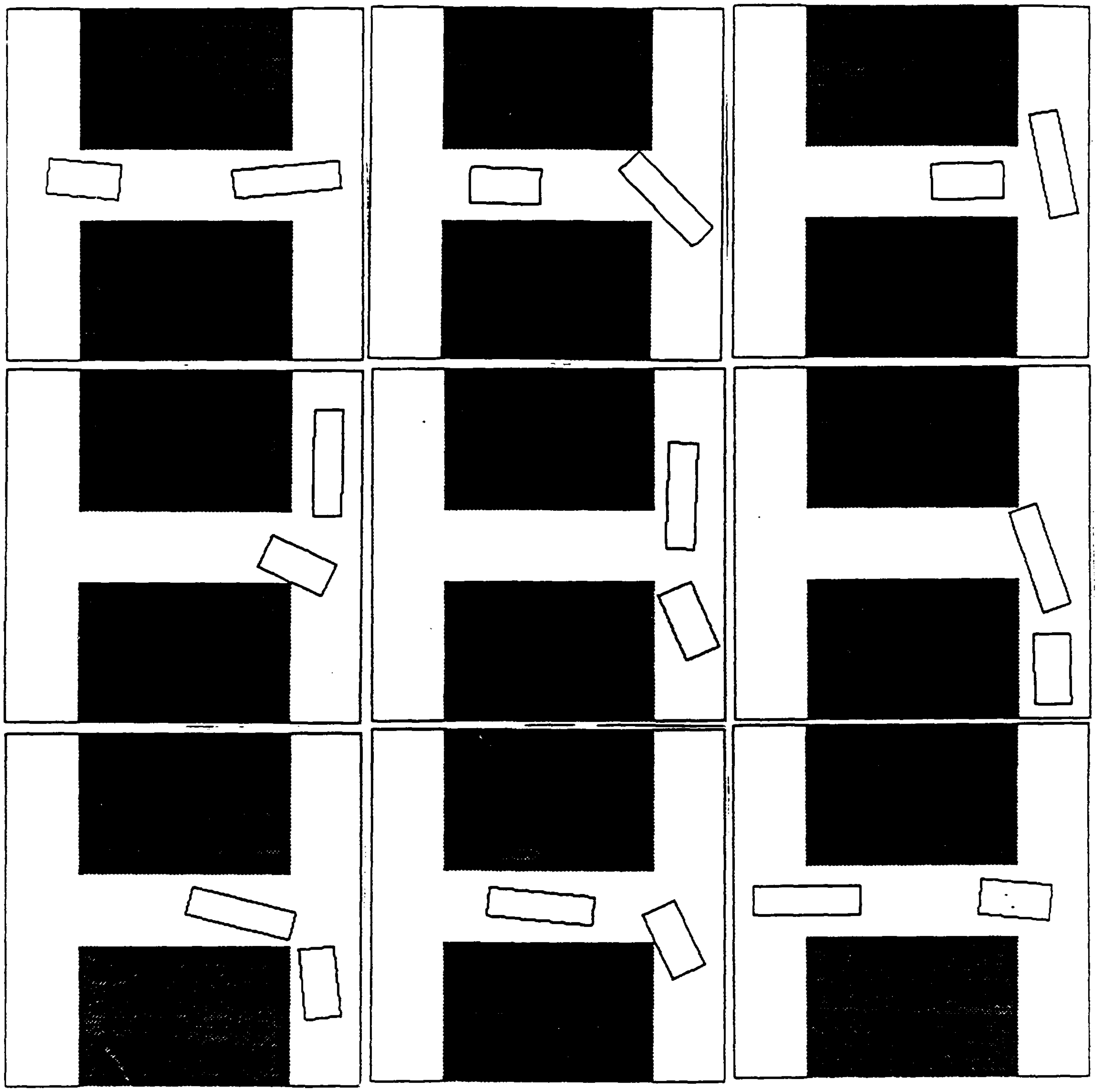

Figure 21: Successive configurations during a more difficult motion (6 DOF multirobot).

neighborhood B of the global minimum containing no other local minimum. As we generate arbitrarily many and arbitrarily long Brownian motions, the probability that the process attains B converges towards 1 by equation (14). But then the process becomes a gradient procedure which reaches immediately the global minimum.

We now prove the validity of statement 14. It is easily proven if $w(t)$ is a 1-dimensional Wiener-Levy process by using the *Désiré André reflection principle* [Papoulis 65]. Its generalization to any connected freespace $\mathcal{C}_{free}$ of any compact manifold $\mathcal{C}$ equipped with a volume measure[8] μ is still true, but based on a deep result about the asymptotic behavior of $w(t)$.

It is important to notice that the compactness assumption for $\mathcal{C}$ is crucial. If $\mathcal{C}$ is not compact, the hypothesis has been proven to be false for $n > 2$ [Doob 84]. As a matter of fact, if $\mathcal{C}_{free}$ is simply equal to $\mathbf{R}^n$, we have the following result [Doob 84]:

Theorem 2 (Hitting of a ball B by a Brownian motion in $\mathbf{R}^n$)

Let w be a Brownian motion in $\mathbf{R}^n$ and B a non-empty ball not containing the starting point of w.

If $n \leq 2$,

$$\lim_{t \longrightarrow \infty} P(M_B(t)) = 1$$

If $n \geq 3$,

$$\lim_{t \longrightarrow \infty} P(M_B(t)) < 1$$

Now, let us consider a connected open subset $\mathcal{C}_{free}$ of a compact manifold with smooth boundaries, equipped with a volume measure μ (In our case, the measure μ is simply $\mu = \frac{1}{\Delta_1}dq_1 \times \ldots \times \frac{1}{\Delta_n}dq_n$.) It can be shown ([Schuss 80]) that the probability distribution of a μ-standard Brownian motion with reflective boundaries starting at any given point q_{INIT} is the solution of the *heat equation*[9]:

$$\forall (q,t) \in \mathcal{C}_{free} \times \mathbf{R}^+, \qquad \frac{\partial p}{\partial t} = \frac{1}{2}\vec{\nabla}^2 p, \qquad p(q,0) = \delta(q - q_{INIT})$$

with Neumann boundary conditions:

$$\forall (q,t) \in \partial\mathcal{C}_{free} \times \mathbf{R}^+, \qquad \frac{\partial p}{\partial \nu} = 0$$

where ν is the normal vector to the boundary. It can also be shown ([Schuss 80]) that this time-varying probability distribution converges towards a stationary distribution $p_\infty(q)$. Therefore, p_∞ is the solution of the following problem:

$$\forall q \in \mathcal{C}_{free}, \qquad \vec{\nabla}^2 p_\infty = 0$$

[8]A volume measure over a compact topological space is a measure which is finite and non-zero for any non-empty open subset

[9][Schuss 80] only treats the case of Euclidean space, but its generalization to manifolds is straightforward [Kendall 87]

$$\forall q \in \partial\mathcal{C}_{free}, \qquad \frac{\partial p_\infty}{\partial \nu} = 0$$

and

$$\int_{C_{free}} p_\infty(q)d\mu(q) = 1$$

But the unique solution of this problem is the μ-uniform distribution:

$$p_\infty(q) = \frac{1}{\mu(C_{free})}$$

As the μ-standard Brownian motion is time-homogeneous, we can apply recursively the Markov property to infer (14).

Finally, we can state:

Monte-Carlo procedure for robot motion planning

Let U be a differentiable real-valued function over an open subset C_{free} of a compact manifold C equipped with a volume measure μ. We assume that the boundary of C_{free} is smooth. We assume that U possesses a unique global minimum q_{GOAL}, and that the number of local minima of U is finite.

We assume that from any point q in C_{free} laying in the attractor of a given local minimum q_{loc} the gradient procedure which leads from q to q_{loc} can be performed in a finite amount of time $T_{GRAD}(q)$. We will denote this procedure by GRADIENT(q,t).

We denote by RANDOM(q,t) the μ-standard Brownian motion process with reflective boundaries starting at $q \in C_{free}$.

We consider the following process $Q(t)$ starting from any configuration $Q(0) = q_{INIT}$.

```
Compute $Q(t) = GRADIENT(q_{INIT}, t)$ from $t = 0$
until we reach a local minimum $q_{cur} = Q(T_{GRAD}(q_{INIT}))$.
Set $t_{cur} = T_{GRAD}(q_{INIT})$.

begin loop
      begin loop
            Generate a random time $T_{RAND}$ following
            the distribution given in formula (13).

            Compute $Q_{candidate}(t) = RANDOM(q_{cur}, t)$ from $t_{cur}$
            until the time $t_{aux}$ for which either of the two conditions
            $U(Q_{candidate}(t)) < U(q_{cur})$ or $t = t_{cur} + T_{RAND}$ is true.

            The final point of the random procedure is
            denoted $q_{aux} = Q_{candidate}(t_{aux})$.
            Compute $Q_{candidate}(t) = GRADIENT(q_{aux}, t)$ from $t_{aux}$ to
            $t_{new} = t_{aux} + T_{GRAD}(q_{aux})$
            where another local minimum $q_{new} = Q_{candidate}(t_{new})$ is reached.

            if $U(Q_{candidate}(t_{new})) < U(q_{cur})$ then exit

        end loop
        Set $q_{cur} = q_{new}$ and $t_{cur} = t_{new}$.
        Concatenate the process $Q(t)$ computed so far with $Q_{candidate}(t)$.

end loop
```

The process Q weakly converges towards the global minimum x_{min} of U. In other words, for any μ-measurable neighborhood B of x_{min}, we have:

$$\lim_{t \longrightarrow \infty} P(Q(t) \in B) = 1$$

This procedure performs a random search of the graph of local minima of the potential field in a best-first fashion, using as heuristic the potential field itself. It behaves very differently from the simulated annealing procedures [Kirk 83], [Geman 84], which perform a breadth-first search of the local minima.

Basically, all simulated annealing procedures can be shown to behave similarly to the following process:

Simulated annealing procedure for optimization

Let U be a differentiable function over $[0,1]^n \subset \mathbf{R}^n$ with a single global minimum x_{min}. Let w be a standard Brownian motion (for the Lebesgue measure) with reflective boundaries in $[0,1]^n$. Let $T(t)$ be a non-negative function converging towards 0 when $t \longrightarrow \infty$ such that for some constant K and for any $t \geq t_0$ the following inequality holds:

$$T(t) \geq \frac{K}{\log t}$$

Consider the solution X of the following stochastic differential equation[10]:

$$dX(t) = -\vec{\nabla} U(X(t))dt + \sqrt{2T(t)}dw(t), \qquad X(0) = x_0 \tag{15}$$

If K is large enough, the process $X(t)$ weakly converges towards x_{min}.

The proof of this result can be found in [Geman 84].

[10] Here, the differentials have to be taken as Itô differentials [Ikeda 81]

The logarithmic rate for the temperature is extremely slow in practical applications. Most of the authors applying this technique choose a much faster annealing schedule (in general an exponential one), in which case the convergence result collapses. In the case of robot motion planning, experimental results show that a classical annealing procedure in not practical. In fact, our Monte-Carlo approach can be considered as a particular case of the diffusion process defined by (15) where the temperature coefficient $T(t)$ is chosen to be 0 during the gradient procedure and to $+\infty$ during the random motions. Instead of choosing a slowly decreasing annealing schedule, we perform a sequence of temperature switches between 0 and $+\infty$. The switches are performed when a local minimum is reached.

The diffusion processes of the form (15) are called *Smoluchowski-Kramers* processes, and their probability distribution $p(x,t)$ is the solution of a partial differential equation called *forward Kolmogorov equation*[11] [Schuss 80]:

$$\frac{\partial p}{\partial t} = L(p), \qquad p(x,0) = \delta(x - x_0)$$

where L is the *Fokker-Planck* operator:

$$L(p) = \vec{\nabla}.(p\vec{\nabla}U) + T\vec{\nabla}^2 p$$

The study of the convergence rate of such processes then reduces to the study of the smallest (in absolute value) non-zero eigenvalue λ_1 of the Fokker-Planck operator. [Matkow 81] and [Schuss 80] give approximate formulas for computing λ_1 in some simple cases. However, in most cases, λ_1 can only be computed numerically, and the only clue for estimating the convergence rate of general diffusion processes is computer simulation.

7 Some Remarks on Parallel Implementations

Thanks to the distributed data representation chosen, the algorithm above is highly parallelizable.

First, the hierarchical decoupled decomposition of the problem allows to compute in parallel all the differents levels of hierarchy, instead of performing sequentially a coarse-to-fine approach. This parallelization process is straightforward, and there is no need to detail it further.

Then, for each resolution level chosen, the planning technique can be roughly divided into three decoupled steps. The first step consists of computing in workspace a few numerical potentials (the distance potential, and the attractive potential fields without local minima) using the wavefront expansion technique described in section 3. The second step consists of performing an incremental best-first search of the connectivity graph built from the set of local minima of the overall configuration space potential field U. The third step consists of smoothing the path obtained by solving a classical two points boundary value problem.

[11] If $U = 0$ and $T = 1/2$ this reduces to the heat equation

We describe respectively in the three subsections below the parallelization issues relative to these three decoupled phases.

7.1 Wavefront Expansion in Workspace

The basic idea underlying all workspace computations in our model is to perform a wavefront expansion of the information. In the case of the distance to the obstacles d_1, the wavefront is initialized to 0 at the obstacles boundaries. In the case of the attractive potential fields, the wavefront is initialized to 0 at the goal point. Then, at each iteration of the expansion, the current wavefront **oldfront** is transformed in the expanded front **newfront**.

There are two basic ways for parallelizing a wavefront expansion. The first, and theoretically the most efficient way, is to allocate one virtual processing unit to each of the members of the front. But two members of the current front **oldfront** may interfere when they are expanded because they may produce identical members of the next front **newfront**, and in fact they do interfere systematically for any non-planar geometrical wavefront. Therefore, the processing units need to *share* memory information. It is well-known that shared-memory parallelism cannot be performed efficiently in a large scale, as it is necessary for our problem. Therefore, the only solution remaining is the so-called "data-parallelism", which consists of affecting to each point of the workspace bitmap a virtual processing unit. In this process, each virtual processor will have its own memory and will only need to communicate with its geometrical neighbors. Parallel architectures such as to one used in the Connection Machine [Hillis 85] are particularly well-suited for this type of application. However, at each cycle time, only those processors which happen to be part of the current wavefronts **oldfront** and **newfront** are actually active. Therefore, this brute force parallelism wastes a lot of parallelism power. We believe nevertheless that this approach is the only realistic way to envision parallelism for geometrical problems.

More precisely, let us consider an SIMD machine with distributed memory having a capability of local communication with 2D or 3D geometrical neighbors. This is typically the case for the Connection Machine [Hillis 85]. We suppose that this machine has $NBPIX = 2^{16} = 65,536$ processors.

We consider a workspace of dimension $n = 2$ or 3 at resolution $r = -\log_2 \delta = 8$, each coordinate having therefore $DIM = 1/\delta = 256$ discrete values. The total number of workspace bitmap points is DIM^n, that is 65,536 in 2D and 16,777,216 in 3D. Therefore, the number $RATIO$ of virtual processors (i.e. of bitmap points) that we have to associate to each physical processor is:

$$RATIO = \max(1, \frac{DIM^n}{NBPIX})$$

This ratio is 1 in 2D and 256 in 3D.

The active wavefront at each time of the expansion is an $(n-1)$-dimensional subset of the n-dimensional workspace. For a typical workspace with a few obstacles, the total number of points in this active wavefront is $COEF \times DIM^{n-1} = 4 \times DIM^{n-1}$. In 2D, this amounts to 1024. In 3D, it is 262,144.

Let us denote by T_s is the computation time that would be required to perform the expansion using a single serial processor. We assume that the number of physical processors is always less than the number of virtual processors, that is $RATIO = DIM^n / NBPIX$. The parallel computation time T_p is approximately (neglecting the communication time between processors, which is realistic in this case):

$$T_p = RATIO \times \frac{T_s}{COEF \times DIM^{n-1}} = \frac{DIM}{COEF \times NBPIX} T_s$$

The above formula leads to two very important conclusions:

1) As long as the number of physical processors is less than the number of virtual processors, ***the parallelism speedup of a wavefront expansion does not depend on the dimension of the workspace.***

2) To obtain an actually significant speedup with distributed memory data-parallelism, the number of physical processors has to be much larger than the number of discrete points representing each coordinate of the workspace.

We obtain the same speedup for a 2D or a 3D workspace:

$$T_p = \frac{T_s}{1024}$$

If the power of each processor in the parallel machine is 100 times smaller than the power of classical workstations (this is approximately the case for current data-parallel machines), we finally get a total speedup of one order of magnitude. As we are able to perform a 2D wavefront expansion on a workstation in about 1/4 seconds, the same computation could theoretically be performed in about 1/40 seconds on a data-parallel machine. For a 3D expansion of the same size, we will need 256 times more time, that is approximately 1 minute on a worksation and 6 seconds on a data-parallel machine.

Finally, we can conclude that a speedup of more than one order of magnitude can only be made on a general purpose SIMD computer only if considerable improvements are made in the hardware.

Therefore, we believe that a more realistic solution for workspace computations is to design a dedicated chip for geometrical wavefront expansion. As the performances of such dedicated systems highly depend on several technological choices, it is not yet possible to give an accurate estimate of the final speedup. We think that real-time capabilities (i.e. a fraction of a second) are achievable for 3D workspaces with current microprocessor technology.

7.2 Parallelizing the Search in Configuration Space

The exit procedure used to escape from a given local minimum q_{loc} consists of generating several random motions from the local minimum. All these random motions can be performed in parallel: We may affect to each of them a different processor, all of these working with a shared memory. The processor which will be the first to find a new local minimum better than q_{loc} will stop all the other processors. This parallelization is straightforward, and there is no need to detail it further. For complex manipulator arms like the one described in section 6.5, the number of random motions necessary the exit from a local minimum can be of several tens for some difficult examples. As the computation time required for the gradient motions is neglectable compared to the time required for exiting a local minimum on a sequential machine, the speedup obtained through this parallelization may be of several tens. For example, in the setting shown in figure 17, the total computation time could decrease from a few minutes to a few seconds by parallelizing the random motions generation.

During this search in configuration space, the potential field U as well as the distance D_1 of the robot to the obstacles have to be computed for each configuration explored. The computation of D_1 involves all the points of the robot body, without interaction between the different points. This is yet another opportunity for parallelism. As a matter of fact, we may affect to each point p on the robot body a dedicated processor computing the forward kinematics $q \longrightarrow X(p,q)$ of this point in the workspace, and communicating with the workspace bitmap processing unit to output the values of the workspace potential field and the distance to the obstacles d_1. Then, the results of all these parallel computations have to be joined together (through computations of minima and maxima) in order to obtain the final overall potential U and the distance D_1. Parallel computers are very efficient for computing non-linear functions such as the maximum or minimum of the output of all the processors. For complex manipulators arms operating nearby obstacles as it is the case for the setting shown in figure 17, the resulting speedup could be of two orders of magnitude (we need about 100 control points when the arm is close to the obstacles). As this last parallelization can be combined multiplicatively with the parallelization of the random motions generation, the total speedup achievable during the configuration space search is of 3 to 4 orders of magnitude. In the example shown in figure 17, the total computation time could fall from 3 minutes to 1/40 seconds with processors equivalent to those available on current workstations. Of course, this impressive speedup would require a very high number of processors (about 10000) of the same power than a current workstation. But even if only 100 processors are available, it is still thinkable to distribute 10000 virtual units over 100 physical processors, so that the speedup would still be of two orders of magnitude.

7.3 Concurrent Computation of Two-Points Boundary Value Problems

Smoothing the path obtained from the Monte-Carlo procedure is a classical quadratic two-points boundary value problem. On a parallel computer, it is possible to affect one virtual processor to each discrete point on the path. As we only consider first derivatives in our objective functional, each processor only needs to communicate with its two neighbors: its predecessor and its successor. Many algorithms are available to optimize in parallel a quadratic functional. The simplest possible is the Jacobi algorithm (i.e. steepest descent gradient applied to a quadratic function). Fancier algorithms like preconditioned conjugate gradient can also be implemented efficiently on a data-parallel machine. In all cases, recent experiments [SRI 89] show that impressive speedups are obtained through the parallelization of such algorithms.

In conclusion, we believe that a dramatic speedup can be achieved in the near future for this Robot Motion Planning algorithm, either by implementing it on a large scale data-parallel machine, or by designing a dedicated parallel architecture. This would lead to real-time performances for robots with a small number of DOF, and to time delays of a few seconds only for complex manipulators with many DOF.

8 Conclusion

In this paper, we have presented a distributed representation approach to path planning applicable for any problem in open freespace. The same approach can be used for mobile robots (with/without non-holonomic constraints) and for arbitrary-shaped manipulator arms.

The approach essentially consists of discretizing both the workspace and the configuration space of the robot, and performing a best-first search of a path using powerful numerical potential fields.

The approach has solved 3 DOF robot problems solved by previous planners, but several orders of magnitude faster (about one second, against several minutes or several tens of minutes for previous planners). More importantly, it has solved problems that fall far outside the range of possibilities of any other existing planners, e.g. 3 DOF robot problems with non-holonomic constraints, 10 DOF manipulator robot problems, and multi-robot cooperation problems.

The algorithm is highly parallelizable. We envision a VLSI implementation which should allow to perform real-time path planning. This opens new perspectives on some key issues in robotics related to the interaction of planning and execution in partially known and dynamically changing environments.

Current work is conducted along several directions. First, we plan to implement the approach on robots operating in 3D workspaces. Second, we intend to experiment our planner on robots having several tens of DOF in order to understand clearly its eventual limitations. Third, we plan to extend this approach to motion planning in partially known environment. Fourth, we will implement this approach on real robots in order to study the new capabilities induced by very fast motion planning. Fifth, we intend to investigate possible generalizations of the distributed representation approach to motion planning and control with uncertainty in control and sensing for part mating and dextrous manipulation operations. As a matter of fact, we believe that this distributed representation of the world can be extended in order to include not only positions, but also velocities, accelerations, internal and external forces, and uncertainty.

References

[Anderson 76] R.F. Anderson & S. Orey. (1976). *Small Random Perturbations of Dynamical Systems with Reflecting Boundary.* Nagoya Math. J., 60(1976), pp. 189-216.

[Barnard 88] S.T. Barnard (1988). *Stochastic Stereo Matching Over Scale* International Journal of Computer Vision, 2 (4), 1988.

[Barra 88] J. Barraquand (1988). *Markovian Random Fields in Computer Vision: applications to seismic data understanding.* Ph.D. Dissertation, Department of Computer Vision, I.N.R.I.A., Sophia-Antipolis, (France) (In French).

[Barra 89] J. Barraquand, B. Langlois, J.C. Latombe (1989). *Robot Motion Planning with Many Degrees of Freedom and Dynamic Constraints.* International Symposium of Robotics Research, Tokyo, Japan (August).

[Boisson 88] J.D. Boissonnat, Olivier Monga (1988). *Scene Reconstruction from Rays. Application to Stereo Data.* IEEE Int. Conf. on Robotics and Automation, Philadelphia.

[Brooks 83] R.A.Brooks, T.Lozano-Pérez (1983). *A Subdivision Algorithm in Configuration Space for Find-Path with Rotation.* Proc. of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe, FRG.

[Canny 87] J.F. Canny (1987). *The Complexity of Robot Motion Planning.* Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT.

[Cerny 85] V. Cerny (1985). *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm.* J. of Optimization Theory and Applications, Vol. 45, 1, January 1985.

[Donald 84] B.R.Donald (1984). *Motion Planning with Six Degrees of Freedom.* Technical Report 791, Artificial Intelligence Laboratory, MIT, Cambridge (May).

[Donald 87] B.R.Donald (1987). *A Search Algorithm for Motion Planning with Six Degrees of Freedom.* Artificial Intelligence, Vol. 31, No. 3, pp. 295-353.

[Doob 84] J.L. Doob (1984). *Classical Potential Theory and Its Probabilistic Counterpart.* Springer-Verlag, 1984.

[Fav 84] B. Faverjon (1984). *Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator.* Proc. of IEEE Conf. on Robotics and Automation, Atlanta.

[Fav 87] B.Faverjon, P.Tournassoud (1987). *A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom.* Proc. of the IEEE International Conference on Automation and Robotics, Raleigh, NC, pp 1152-1159.

[Feller 66] W. Feller (1966). *An Introduction to Probability Theory and its Applications.* Vol. 1 and 2, John Wiley, New-York, 1966.

[Fernique 67] X. Fernique (1967). *Processus Linéaires, Processus Généralisés* Ann. de l'Inst. Fourier, Grenoble, XVII, Fasc. 1, 1967 (In French).

[Geman 84] D. & S. Geman (1984). *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 6, pp. 721-741, 1984.

[Geman 86] S. Geman & C.H. Hwang (1986). *Diffusions for Global Optimization* SIAM Journal on Control and Optimization, Vol. 24 No 5, September 1986.

[Gouzenes 84] L.Gouzènes (1984). *Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots.* International Journal of Robotics Research, Vol. 3, No. 4.

[Hillis 85] D. Hillis (1985). *The Connection Machine.* MIT Press 1985.

[Ikeda 81] N. Ikeda & S. Watanabe. (1981). *Stochastic Differential Equations and Diffusion Processes.* North-Holland Mathematical Library, 1981.

[Kendall 87] W. S. Kendall. (1987). *Stochastic Differential Geometry: An Introduction.* Acta Applicandae Mathematicae, Vol. 9, pp 29-60.

[Khatib 86] O. Khatib (1986). *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.* Intern. Journal of Robotics Research 5, 1, MIT Press.

[Kirk 83] S. Kirkpatrick, C.D. Gelatt, & M.P. Vecchi. (1983) *Optimization by Simulated Annealing.* Science, Vol. 220, pp 671-680.

[Kod 87] D. E. Koditschek. (1987) *Exact Robot Navigation by means of Potential Functions: Some Topological Considerations.* IEEE Int. Conf. on Robotics and Automation, Raleigh, NC.

[Laumond 86] J.P. Laumond (1986).*Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints.* Proc. Int. Conf. on Intelligent Autonomous Systems, Amsterdam (Dec. 1986) pp 346-354.

[Laumond 87] J.P. Laumond (1987). *Finding Collision Free Smooth Trajectories for a Non-Holonomic Mobile Robot.* Proc. Int. Conf. on Artificial Intelligence, Milano, August 1987.

[Lozano 87] T.Lozano-Pérez (1987). *A Simple Motion-Planning Algorithm for General Robot Manipulators.* IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, pp. 224-238.

[Lumelsky 87] V.Lumelsky (1987). *Algorithmic Issues of Sensor-Based Robot Motion Planning.* Proc. of the 26th IEEE Conference on Decision and Control, Los Angeles, pp. 1796-1801 (December).

[Matkow 81] B.J. Matkowsky & Z. Schuss (1981). *Eigenvalues of the Fokker-Planck Operator and the Approach to Equilibrium for Diffusions in Potential Fields.* SIAM J. Appl. Math., Vol 40, pp 242-254, 1981.

[Papoulis 65] A. Papoulis (1965). *Probability, Random Variables, and Stochastic Processes.* Mc. Graw-Hill 1965.

[Rimon 88] E.Rimon, D.E.Koditschek (1988). *Exact Robot Navigation using Cost Functions: The Case of Distinct Spherical Boundaries in E^n.* Proc. of the IEEE International Conference on Robotics and Automation, Philadelphia, pp. 1791-1796.

[Reif 79] J.H.Reif (1979). *Complexity of the Mover's Problem and Generalizations.* Proc. of the 20th Symposium on the Foundations of Computer Science, pp. 421-427.

[Reif 85] J. Reif, J. Storer (1985). *Shortest Paths in Euclidean Space with Polyhedral Obstacles.* Technical report CS-85-121, Computer Science Department, Brandeis University (April).

[Roubine 70] E. Roubine. (1970) *Introduction à la Théorie de la Communication.* Vol 1 to 3. Monographies d'Electronique, Masson, Paris 1970. (In French).

[Schuss 80] Zeev Schuss (1980). *Theory and Appli ations of Stochastic Differential Equations.* Wiley and Sons, New-York, 1980.

[Schwartz 82] J.T.Schwartz, M.Sharir (1982). *On the Piano Movers Problem: II. General Techniques for Computing Topological Properties of Algebraic Manifolds.* Technical Report 41. New York University Computer Science Department, Courant Institute of Mathematical Sciences.

[Schwartz 86] J.T.Schwartz, M.Sharir (1986). *A Survey of Motion Planning and Related Geometric Algorithms.* Proc. of Oxford Workshop on Geometric Reasoning, Oxford, England.

[Schwartz 87a] J.T.Schwartz, M.Sharir, J.Hopcroft (1987). *Planning, Geometry, and Complexity of Robot Motion.* Ablex, Norwood, New Jersey.

[Schwartz 87b] J.T.Schwartz,C.K.Yap (1987). *Algorithmic and Geometric Aspects of Robotics.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[Spivak 79] M. Spivak (1979). *A Comprehensive Introduction to Differential Geometry.* Vol. 1 to 5, Publish or Perish. Berkeley 1979.

[SRI 89] M. A. Fischler, O. Firschein, S.T. Barnard, P. Fua, Y. Leclerc (1989) *The Vision Problem: Exploiting Parallel Computation* Technical Note No 458, SRI International, Menlo Park, California, February 1989, and Fourth Int. Conf on Supercomputing, Santa Clara, California, May 1989.

[Tourn 88] P. Tournassoud (1988) *Motion Planning for a Mobile Robot with a Kinematic Constraint.* IEEE Int. Conf. on Robotics and Automation, Philadelphia, April 1988.