CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Contingency-Tolerant

Robot Motion Planning and Control

Wonyun Choi, David Zhu
and
Jean-Claude Latombe

(To be presented at the IEEE/RSJ International Workshop on
"Intelligent Robots and Systems" IROS/89, Tsukuba, Japan, 9/4-6/'1989)

TECHNICAL REPORT
Number 14

July, 1989

Contingency-Tolerant Robot Motion Planning and Control

Wonyun Choi, David Zhu, Jean-Claude Latombe
Robotics Laboratory
Department of Computer Science, Stanford University
Stanford, CA 94305, USA

Abstract: We address the problem of robot motion planning and control in a partially known environment. Examples of this type of environment include shop-floors, office buildings, construction sites, and clean rooms. In such environments, the shapes and the locations of the largest objects are known in advance. But there are other objects whose locations are changing so often that the robot cannot realistically keep track of them. In order for a robot to operate successfully in this type of environment, it must be tolerant to contingencies – i.e., it must be able to efficiently deal with unexpected obstacles while executing planned motions. A contingency-tolerant motion planning and control system is presented in this paper. It combines a "lesser-commitment" planner with an "intelligent" controller. The planner produces a set of paths, called a "channel", rather than a single path, in order to let the controller have more freedom of choice. The controller exploits this freedom by applying a potential field method. We have implemented this system and experimented with it, using both a computer simulated mobile robot and a real one.

Acknowledgements: This research was funded by DARPA contract DAAA21-89-C0002 (Army), CIS (Center for Integrated Systems), CIFE (Center for Integrated Facility Engineering), and Digital Equipment Corporation.

1 Introduction

Most of the proposed robot motion planning/control systems are making either the assumption that the workspace is completely known to the robot at planning time or the assumption that it is completely unknown. For systems that assume complete prior knowledge, the planner generates a single path and the controller makes the robot follow the path [11] [15]. For systems that assume no prior knowledge, the controller drives the robot using only local information obtained from the sensors [8] [12].

In most applications, none of these two extreme assumptions is satisfied. Except for some specific exploratory tasks, it is always possible to provide the robots with some prior knowledge about their workspace. On the other hand, in most cases, this knowledge is partial because it is too expensive, or even impossible, to maintain complete knowledge of the workspace, especially a dynamically changing one.

In this paper, we address the problem of robot motion planning and control in a partially known workspace. Typical application examples include automated transportation tasks in manufacturing shop-floors, office environments, civil engineering construction sites, and clean rooms. In these environments, the shapes and the locations of the largest objects such as machines, furnitures, and walls are known to the robot at planning time, but there are other objects such as human beings and other moving or easily movable small objects, whose locations are changing so often that the planner cannot realistically keep track of them.

The approach to motion planning and control which consists of generating a single path and then making the robot follow this path, is inadequate when prior knowledge of the workspace is incomplete. Indeed, if the robot detects unexpected obstacles while executing the planned path, it will stop and will not know what to do next, without, say, backtracking to the planner. The failure to deal with contingencies at a low level is due to the fact that a path is an overly constrained plan. On the other hand, systems that assume no prior knowledge and control the robots using sensory feedback are grossly inefficient due to the lack of global knowledge. In particular they may easily get stuck into obstacle concavities.

We believe that increased robustness and efficiency can be achieved by integrating "lesser-commitment" planning with "intelligent" control. A lesser-commitment motion planner produces a set of paths in order to let the controller have more freedom of choice. To take advantage of this freedom, an intelligent control system must have the ability to make decisions at execution time based on its perception of the workspace. Based on this philosophy, we have developed an integrated motion planning and control system in which the planner generates not a single path but a set of contiguous paths – a *channel* – and the controller constrains the robot to move within the channel. Provided with a channel, the controller has the necessary global information to guide its motion in the proper direction. It also has enough flexibility to deal with unexpected obstacles by adapting its motion within the channel. We have implemented this approach and experimented with it, using both a computer simulated mobile robot and a real mobile robot called $\mathcal{GOFER}$.

2 Overview of the Approach

Let us denote by $\mathcal{A}$ the robot and by $\mathcal{W}$ the workspace. We attach a Cartesian frame $\mathcal{F}_\mathcal{A}$ to $\mathcal{A}$ and a Cartesian frame $\mathcal{F}_\mathcal{W}$ to $\mathcal{W}$. A configuration q of $\mathcal{A}$ is a specification of the position and orientation of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$. The configuration space of $\mathcal{A}$, denoted by $\mathcal{C}$, is the set of all the possible configurations of $\mathcal{A}$. The subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration q is denoted by $\mathcal{A}(q)$.

Throughout the paper, we model the robot as a two-dimensional object moving in a two-dimensional space $\mathcal{W}$ isomorphic to $\mathbb{R}^2$. Therefore, $\mathcal{C}$ is isomorphic to either $\mathbb{R}^2$ (if $\mathcal{A}$ can only translate or if it is a disc) or $\mathbb{R}^2 \times S^1$, where S^1 is the unit circle (if $\mathcal{A}$ can both

translate and rotate). However, since the concepts underlying our approach are more general[1], we keep our presentation as independent as possible from these assumptions.

In addition to $\mathcal{A}$, $\mathcal{W}$ contains obstacles denoted by $\mathcal{B}_i$ ($i = 1$ to n). All of them are assumed to be stationary. Each obstacle $\mathcal{B}_i$ maps in $\mathcal{C}$ to the region $\mathcal{CB}_i = \{q \in \mathcal{C} \ / \ \mathcal{A}(q) \cap \mathcal{B}_i \neq \emptyset\}$, which is called a C-obstacle. The complement to the union of all the C-obstacles in $\mathcal{C}$ is called the free space and is denoted by $\mathcal{C}_{free}$. A free path of $\mathcal{A}$ from an initial configuration q_{init} to a goal configuration q_{goal} is a continuous map $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, such that $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$.

Let us now suppose that the position and geometry of the obstacles $\mathcal{B}_1$ through $\mathcal{B}_q$, $q \leq n$, are known at planning time, while no information is available about the other obstacles, if any. We write $\mathcal{C}'_{free} = \mathcal{C} - \bigcup_{j=1 \text{ to } q} \mathcal{CB}_j$. Obviously: $\mathcal{C}_{free} \subseteq \mathcal{C}'_{free}$.

We are interested in planning and controlling the motion of $\mathcal{A}$ along a free path connecting q_{init} to q_{goal} among all the obstacles $\mathcal{B}_1$ through $\mathcal{B}_n$. We assume that $\mathcal{A}$ is instrumented with N proximity sensors for detecting "unexpected" obstacles[2]. Our approach to this problem is the following:

1. At planning time, $\mathcal{C}'_{free}$ is treated as the actual free space. Rather than generating a path between q_{init} to q_{goal} in $\mathcal{C}'_{free}$, we extract a connected subset Π of $\mathcal{C}'$ containing both q_{init} and q_{goal}. This region, called a "channel", should ideally possess the following two properties[3]:
   - *P1*: It is possible to compute an artificial potential field U_0 over Π, for which q_{goal} is a stable equilibrium state whose domain of attraction includes the entire set Π. In addition, U_0 tends to infinity when $\mathcal{A}$'s configuration tends toward the boundary of Π.
   - *P2*: Π is maximal under *P1*.
2. At execution time, the motion of the robot is controlled along a path tangent to $-\nabla U$, with $U = U_0 + \Sigma_{k=1 \ to N} U_k$, where U_k is a repulsing potential field generated from the data provided by the kth proximity sensor. U_k is non-zero only when the sensor detects an unexpected obstacle inside Π. It tends toward infinity when the distance to this obstacle tends toward zero.

This approach can be intuitively justified as follows. By first treating $\mathcal{C}'_{free}$ as the free space, the planner makes use of the available knowledge about the obstacles in an optimistic fashion. This is reasonable since in most practical applications the shapes and locations of the largest obstacles are known at planning time. However, by extracting a channel Π the planner does not impose the robot to follow a particular path which would possibly cross an unknown obstacle. The choice of Π with property *P1* guarantees that in the absence of unknown obstacles – then $\mathcal{C}_{free} = \mathcal{C}'_{free}$ – the potential field will guide the robot to q_{goal} without getting stuck at a local minimum or escaping from the channel. In the presence of unexpected obstacles, the robot will not escape the channel either, since the potential barrier becomes infinite on Π's boundary. Furthermore, thanks to the term $\Sigma_k U_k$ in the expression of the potential field, the robot will not hit unexpected obstacles.

The combination of U_0 with another non-zero term does no longer guarantee that U is free of local minima. $\Sigma_k U_k$ needs to be non-zero only in a small surrounding of the corresponding C-obstacles. Then, if the unexpected obstacles are small enough, we can expect that every local minimum, if any, will have a limited domain of attraction within Π. Nevertheless, in order to make the implemented system more robust, we supplement it with techniques for dealing with local minima.

The central issues in implementing this approach are the construction of the channel and the definition of the potential field. The definition of a potential field over an arbitrary region, with a single stable equilibrium state at the goal configuration, has been studied in [9]. It turns out to be a very delicate issue with no known general solution, although specific ones have been proposed in Euclidean configuration spaces, when all the C-obstacles are spherical objects [13] or star shaped objects [14]. Even if a general solution was established, it is most likely that the corresponding potential fonction would be very costly to compute. In our implemented approach, we turn this difficulty by restricting the possible shapes of Π to a sequence of adjacent non-overlapping rectangloids. With this restriction, we can define a potential field U_0 that both satisfies property *P1* and is quick to compute. A channel constructed as a sequence of rectangloids cannot be maximal under property *P2*. Nevertheless, the simplification seems to be a reasonable compromise between achieving absolute least-commitment at planning time, on the one hand, and efficiently computing a potential function with a single stable equilibrium state at the goal, on the other hand.

3 Channel Generation

3.1 Method

We parameterize a configuration $q \in \mathcal{C}$ by a list of m generalized coordinates $(q_1, ..., q_m)$ in a Cartesian space, where m is the dimension of the configuration space manifold [1]. We assume, without practical loss of generality, that the range of possible values for the q_i's are closed intervals $[q_i^{min}, q_i^{max}]$. Hence, we represent $\mathcal{C}$ as a closed rectangloid:

$$[q_1^{min}, q_1^{max}] \times ... \times [q_m^{min}, q_m^{max}] \subset \mathbf{R}^m.$$

For example, if $\mathcal{C} = \mathbf{R}^2$, we take $q = (x, y)$, with x and y being the coordinates of the origin $O_\mathcal{A}$ of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$. If $\mathcal{C} = \mathbf{R}^2 \times S^1$, we take $q = (x, y, \theta)$, with x and y defined in the same fashion, and θ being the angle (mod. 2π) between the x-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$. We represent $\mathcal{C}$ as $[x^{min}, x^{max}] \times [y^{min}, y^{max}] \times [0, 2\pi]$, with the two faces $\theta = 0$ and $\theta = 2\pi$ procedurally identified.

A channel is a subset Π of $\mathcal{C}'_{free} = \mathcal{C} - \bigcup_{j=1,...,q} \mathcal{CB}_j$, where $\mathcal{CB}_j$, $j = 1, ..., q$, are the known obstacles. It contains both the initial configuration q_{init} and the goal configuration q_{goal}. As mentionned in Section 2, we restrict a channel to a sequence $(\kappa_1, ...\kappa_p)$ of *rectangloid cells* such that:

- $q_{init} \in \kappa_1$ and $q_{goal} \in \kappa_p$;
- $\forall i \in [1, p]$, $int(\kappa_i) \subset \mathcal{C}'_{free}$ – i.e., the interior of every cell is contained in $\mathcal{C}'_{free}$;
- $\forall i, j \in [1, p]$, $i \neq j$, $int(\kappa_i) \cap int(\kappa_j) = \emptyset$ – i.e., no two cells overlap;

[1] For example, the same approach could be applied to a six-degree-of-freedom free-flying platform and to a manipulator arm.

[2] In some sense, we expect the obstacle, otherwise we will not use sensors to detect it. By unexpected, we mean that the location and shape of the obstacle is not known in advance.

[3] Notice that the notion of channel defined here has no apparent relation with the notion of "channel" used in [5] and the notion of "freeway" used in [2].

- $\forall i \in [1, p-1]$, $\kappa_i \cap \kappa_{i+1}$ is a $(m-1)$-dimensional rectangloid – i.e., two successive cells in the sequence are adjacent by sharing a portion of their boundary having non-zero measure in $\mathbf{R}^{m-1}$.

When $\mathcal{C} = \mathbf{R}^2 \times S^1$, a spurious effect of the Cartesian representation of the configuration space is to introduce an artificial boundary for the cells at 0 and 2π. In order to remove this non-justified boundary, we allow a rectangloid cell to be made of the two following regions:

$$[x^{min}, x^{max}] \times [y^{min}, y^{max}] \times [0, \theta_1]$$

and

$$[x^{min}, x^{max}] \times [y^{min}, y^{max}] \times [\theta_2, 2\pi].$$

In the following, we will continue to consider such a cell as a single rectangloid, although it is actually represented in the Cartesian space as two rectangloids.

If $\mathcal{C} = \mathbf{R}^2$, the boundary $\partial\kappa_i$ of a cell κ_i simply consists of the four edges of the cell. If $\mathcal{C} = \mathbf{R}^2 \times S^1$, $\partial\kappa_i$ consists of the six faces of the corresponding rectangloid, if κ_i does not range over all the orientations in S^1. Otherwise, it only consists of the four faces perpendicular to the x and y axes.

The boundary $\partial\Pi$ of the channel $\Pi = (\kappa_1, ..., \kappa_p)$ is defined as:

$$\partial\Pi = \bigcup_{i=1}^{p} \partial\kappa_i \; - \bigcup_{i=1}^{p-1} \kappa_i \cap \kappa_{i+1}.$$

In order to construct a channel, we have adopted a popular approach to motion planning, which is known as *hierarchical approximate cell decomposition* [3]. This approach provides a balance between efficiency and completeness and is relatively simple to implement

The approach consists of decomposing the configuration space of the robot into rectangloid cells at successive levels of approximation. Cells are classified to be EMPTY or FULL, depending on whether their interiors lie entirely outside or entirely inside the obstacles. If they are neither EMPTY, nor FULL, they are labelled MIXED. At each level of approximation, the planner searches the graph of the adjacency relation among the cells for a sequence of adjacent EMPTY cells connecting the initial configuration of the robot to the goal configuration. If no such sequence is found, it decomposes some MIXED cells into smaller cells, label them appropriately, and searches again for a sequence of EMPTY cells. In theory, the process ends when a solution has been found or it is guaranteed that no solution can be found. In practice, due to uncertainty in robot control, it is suitable to impose some minimal size requirement to both the size of the cells in a channel and the size of the intersection of two successive cells in the channel. Imposing a minimal size to cells may also lead the algorithm to terminate more quickly.

In spite of the simplicity of the underlying principles, an efficient implementation of the approach raises delicate algorithmic problems related to cell decomposition and hierarchical graph searching. We found out that efficiency can be sharply increased by shifting from naive solutions to these problems, to more sophisticated ones. This led us to develop new efficient algorithms, which are described in detail in [17]. Experiments have shown that our planner, which is based on these algorithms, is significantly faster than previous planners based on the same general approach. We briefly present these algorithms in the next two subsections.

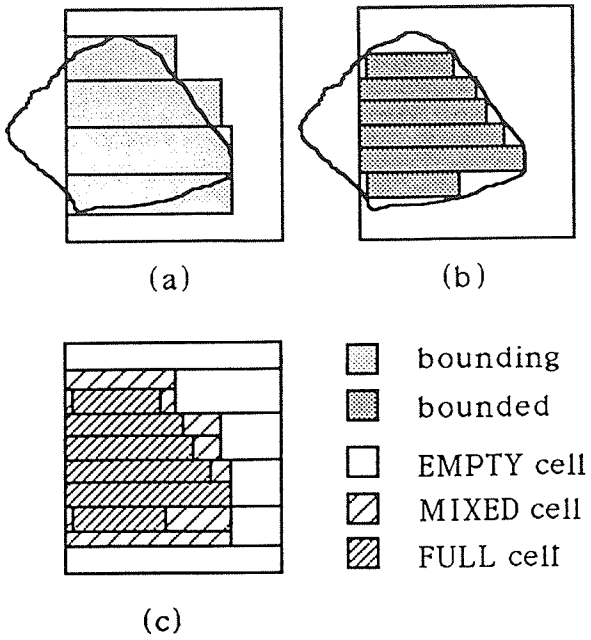

Figure 1: Bounding and Bounded Approximations

3.2 Cell Decomposition

In decomposing a MIXED cell, we wish to achieve two conflicting goals:

- Minimize the number of cells in the decomposition, in order to keep the size of the search graph as tractable as possible.
- Maximize the volume of the EMPTY and FULL cells, in order to find a channel or detect that no channel exists as quickly as possible.

The blind 2^m-tree decomposition ("quadtree" if $m = 2$ and "octree" if $m = 3$) used for example in [6] and [7] has the drawback of decomposing many MIXED cells into smaller MIXED cells. The technique described in [3] seems superior. But, because it treats the C-constraints[4] individually, it still over-fragments the cells, which not only affects the efficiency of the search, but also results in channels that are narrower than necessary.

Our technique consists of first approximating each C-obstacle lying in the MIXED cell to be decomposed as a collection of non-overlapping rectangloids. The complement of a union of rectangloids within a rectangloid region is also a union of rectangloids[5], which can easily be computed. It yields a rectangloid decomposition of the MIXED cell. We call this technique *contraint reformulation*, since it basically consists of reformulating the constraints imposed by the C-obstacles into a form directly compatible with the format of the decomposition of the cell into rectangloids.

Our planner generates and uses *bounding* and *bounded* approximations of C-obstacles, as illustrated in Figures 1a and 1b, respectively. The EMPTY cells of the decomposition of a MIXED cell κ are obtained by computing the complement of the bounding approximation of the C-obstacles in κ. The FULL cells are given by the bounded approximation. The MIXED cells are extracted from the difference between the bounding and the

[4] The C-constraints are the constraints that define the surfaces bounding the C-obstacles.

[5] All the rectangloids have their edges parallel/perpendicular to the axes of the Cartesian space used to represent the configuration space.

bounded approximations. This is illustrated in Figure 1c. Both the bounding and the bounded approximations are generated using a "project-lift" method. Each C-obstacle within κ is first projected onto the xy-plane (for a certain θ interval) and then onto either the x or the y axis (for a certain y or x interval). The projections are intervals along the x or y axis, which are lifted back to rectangles in the xy plane and then to rectangloids in $\mathcal{C}$. (See [17] for details.)

3.3 Graph Search

As described in Subsection 4.1, the process of generating a channel interleaves cell decomposition and graph search. The graph to be searched at each iteration represents the adjacency relation among the current EMPTY and MIXED cells. We call it the **cell-connectivity graph** (ccg for short).

A simple algorithm for this process can be stated as follows. (We call a channel found by the search a *solution channel*, if it contains only EMPTY cells, and a *candidate channel*, if it contains MIXED cells.)

1. Generate an initial decomposition of $\mathcal{C}$ and construct the ccg $\mathcal{CCG}_0$ corresponding to this decomposition. Set counter i to 0.
2. Search the current ccg $\mathcal{CCG}_i$ for a channel[6]. If a solution channel is found, exit with success. If a candidate channel is found, go to step 3. Otherwise – i.e., no channel has been found – exit with failure.
3. Generate a decomposition for every MIXED cell in the candidate channel and generate a new graph $\mathcal{CCG}_{i+1}$ by embedding the decompositions of the MIXED cells into the previous graph $\mathcal{CCG}_i$. Set counter $i = i + 1$. Go to Step 2.

The major drawback of the simple algorithm given above is that the search work performed in $\mathcal{CCG}_i$, if it does not return success, is not used to help the search of $\mathcal{CCG}_{i+1}$. This can be remedied by the following *divide-and-conquer* algorithm. Rather than reconstructing of a full ccg whenever the MIXED cells along a candidate channel are refined, a ccg representing the decomposition of every refined cell is generated separately and recursively searched for a *subchannel*. The new algorithm hence generates a hierarchy of ccg's. The ccg at the top of the hierarchy corresponds to the initial decomposition of $\mathcal{C}$ and is denoted by $\mathcal{CCG}_\mathcal{C}$. Every other ccg corresponds to the decomposition of a certain MIXED cell, say κ, and is denoted by $\mathcal{CCG}_\kappa$. A channel Π, if any, is first generated in $\mathcal{CCG}_\mathcal{C}$. If Π is a solution channel, the planner exits with success; otherwise, each MIXED cell κ in Π is decomposed recursively, and a subchannel Π_κ, if any, is generated in $\mathcal{CCG}_\kappa$. This subchannel is substituted for κ in Π.

In order to make this algorithm work properly and efficiently, we need to work out the following details:

- Each subchannel Π_κ must connect appropriately to the rest of Π [7]. This is done by generating a complete channel connecting $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ at every level of refinement.
- If the search fails to find a subchannel within a MIXED cell, the planner must backtrack to the previous level and search for an alternative channel. It uses dependency-directed backtracking techniques [16] [10] in order to avoid running into the the same mistakes several times.

[6] The search is guided by various types of heuristics, which take into consideration the length of the generated channel and the expected cost of future cell decomposition and graph search. Therefore, although EMPTY cells are preferred to MIXED ones, a candidate channel may be preferred to a solution channel, if the former is substantially shorter.

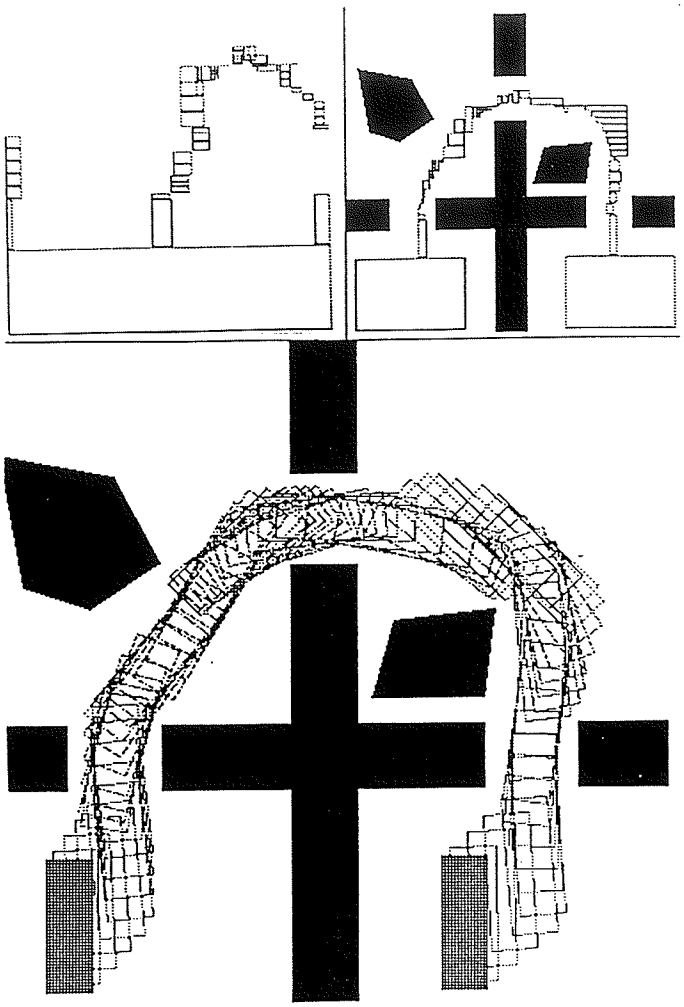

Figure 2: Example of Channel

These "details" are described in length in [17]. Figure 2 shows a channel generated by the planner for a rectangular robot that can both translate and rotate (top) and a path extracted from the channel (bottom). The origin of $\mathcal{F}_\mathcal{A}$ is at the center of the rectangle and its x axis along the major axis of the rectangle. The channel is visualized by its two projections on the xy and $x\theta$ planes. The generation of this channel took approximately 3 minutes on an Apple Macintosh II computer.

4 Robot Control in a Channel

4.1 Method

Our robot controller makes use of a potential field method [8]. This basically means that $\mathcal{A}$ is regarded in the configuration space $\mathcal{C}$ as a unit mass particle moving along the flow of a force field $F(\mathbf{q}) = -\nabla \mathbf{U}(\mathbf{q})$, where $\mathbf{U}$ is the potential function.

The actual dynamic equation of motion of $\mathcal{A}$ in its configuration space is:

$$\Lambda \ddot{\mathbf{q}} + \mu(\dot{\mathbf{q}}) - F_m = 0$$

where Λ is the kinetic energy matrix, $\ddot{q}$ is the robot generalized acceleration, $\mu(\dot{q})$ represents the centrifugal and Coriolis generalized forces, and F_m is the generalized force applied by the actuators. The robot is treated as a unit mass particle moving in U by selecting F_m as the command vector and computing it as:

$$F_m = \Lambda[-\nabla U(q)] + \mu(\dot{q}).$$

As mentioned earlier, the potential function U is defined as the sum of two terms, U_0 and $\sum_k U_k$. The potential U_0, which we call the **channel potential**, guides the motion of $\mathcal{A}$ through the channel Π generated by the planner. The potential $\sum_k U_k$, which we call the **contingency potential**, repulses $\mathcal{A}$ away from the unexpected obstacles detected by the proximity sensors.

4.2 Channel Potential

Let $\Pi = (\kappa_1, ..., \kappa_p)$ be the channel generated by the planner.

The channel potential U_0 should determine a single equilibrium state over Π located at q_{goal}. It should also grow toward infinity near the boundary of Π, so that $\mathcal{A}$ cannot escape Π, even in the presence of unexpected obstacles.

One simple way to define U_0 is to add an attractive potential U_0^g pulling the robot toward the goal configuration and a repulsive potential U_0^w pushing the robot away from the "walls" (i.e., the boundary) of the channel. However, a channel is usually not a convex region, and this simple definition is not acceptable, since it would often result in a function U_0 with local minima other than the goal.

One fashion to proceed is to construct a sequence of intermediate goal configurations in the channel, the last configuration in the sequence being q_{goal}. Then, we can define U_0 so that $\mathcal{A}$ is attracted by each intermediate goal configuration in turn. The issues are (1) how to choose the intermediate goal configurations, and (2) when to shift from one intermediate goal to the next.

A possible sequence of intermediate goals is $(q_1, q_2, ..., q_p)$, where q_i is the midpoint of $\kappa_i \cap \kappa_{i+1}$, for $i = 1, ..., p-1$, and $q_p = q_{goal}$. Then, in each cell κ_i, we can define the potential U_0 as the sum of an attractive potential pulling the robot toward q_i and a repulsive potential pushing it away from the boundary of the channel. If the robot is not damped by adding a dissipative derivative term, it will traverse $\kappa_i \cap \kappa_{i+1}$ with a non-zero velocity. At this instant, the controller will consider q_{i+1} as the new goal. When the last cell κ_p is entered, q_{goal} becomes the goal and the controller damps the motion of $\mathcal{A}$ by adding a dissipative force proportional to the velocity (see [8]), in order to attain and stop at q_{goal} without overshoot.

The problem with the above definition is that the attractive force tends toward zero when the robot gets closer from q_i, while still being in κ_i. Hence, in the vicinity of q_i, the robot is almost completely under the influence of the contingency potential, if it is not zero. This seems likely to increase the risks of creating spurious stable equilibrium states near the intermediate goals. One way to solve this drawback is to make the controller abandon every intermediate goal before it is attained and shift to the next, so that the attractive force never vanishes. This led us to retain a slightly different definition for U_0, which is presented below.

Let us consider a cell κ_i. We call the $(m-1)$-dimensional rectangloid $\kappa_{i-1} \cap \kappa_i$ (resp. $\kappa_i \cap \kappa_{i+1}$), with the convention $\kappa_0 = \kappa_{p+1} = \emptyset$, the access gate (resp. the exit gate) of κ_i.

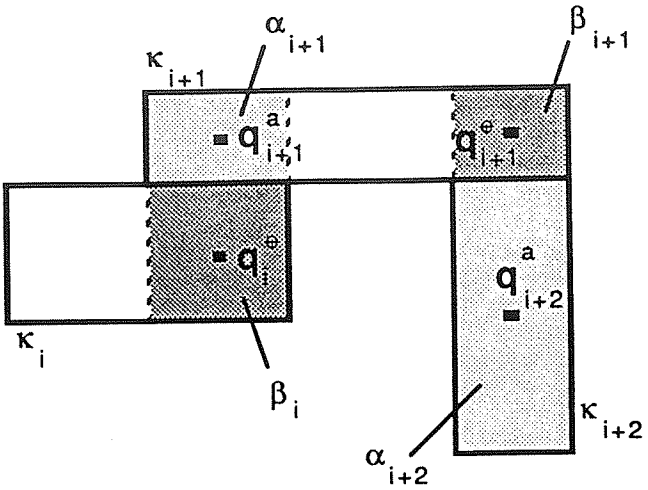

Figure 3: **Intermediate Goals**

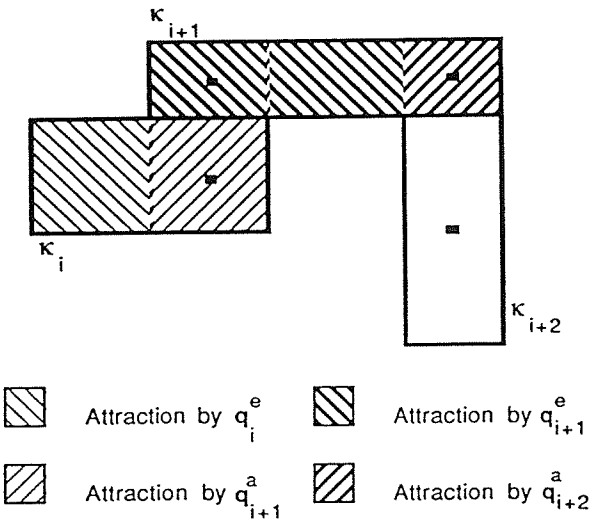

Figure 4: **Shifting Between Goals**

We denote by α_i (resp β_i) the region obtained by sweeping the access gate (resp. the exit gate) of κ_i perpendicularly to itself inside κ_i. Both α_i and β_i may be identical to κ_i. In the first cell of the channel, we only define β_1. In the last cell, we only define α_p. Figure 3 illustrates the construction of the α_i's and the β_i's in a two-dimensional channel.

For every cell κ_i, we define the midpoints of α_i and β_i as two intermediate goal configurations, which we respectively denote by q_i^a and q_i^e. Hence, the sequence of intermediate goals is:

$$(q_1^e, q_2^a, q_2^e, ..., q_{p-1}^e, q_p^a, q_{goal}).$$

These intermediate goals are shown in Figure 3.

We construct U_0 in a piecewise fashion over overlapping rectangloid regions. Each such region is either a *regular rectangloid*, i.e. a cell of the channel, or an *intermediate rectangloid*, i.e. the union γ_i of β_i and α_{i+1}, with $i \in [1, p-1]$. The rectangloid γ_i thus overlaps κ_i and κ_{i+1}. The goal configuration in the regular rectangloid κ_i is q_i^e, if $i \neq p$, and q_{goal}, otherwise. The goal configuration in the intermediate rectangloid γ_i is q_{i+1}^a. U_0 is defined over each rectangloid as the sum of two terms, an attractive

term U_0^g, which pulls the robot toward the goal configuration q_g of the rectangloid, and a repulsive term U_0^w, which pushes the robot away from the boundary of the channel. The controller shifts from one intermediate goal to the next whenever it enters a new (regular or intermediate) rectangloid. For example (see Figure 4), if $\mathcal{A}$'s configuration is in κ_i, $i \in [1, p-1]$, and not in γ_i, the current goal is q_i^c. As soon as it enters γ_i, the current goal becomes q_{i+1}^a. When it enters κ_{i+1}, it becomes q_{i+1}^c, if $i+1 \neq p$, and q_{goal}, otherwise. If $p = 1$, there is no intermediate goal and q_{goal} is immediately taken as the goal to attain.

Shifting from one intermediate goal to the next as explained above results in a discontinuous field of attractive forces. Discontinuities can be smoothed at the servo level. Another technique would consist of shifting continuously from an intermediate goal to the next, by making the goal vary along the segment connecting them.

The potentials U_0^g and U_0^w can be formally defined in several ways. Our definitions are directly drawn from those given in [8]. We take:

$$U_0^g = \frac{1}{2} K_g \rho_g^2(q)$$

and:

$$U_0^w(q) = \begin{cases} \frac{1}{2} K_w (\frac{1}{\rho_w(q)} - \frac{1}{\rho_0})^2 & \text{if } \rho_w(q) \leq \rho_0, \\ 0 & \text{if } \rho_w(q) > \rho_0. \end{cases}$$

where:

- K_g and K_w are scaling factors,
- $\rho_g(q)$ is the distance between q and the current goal q_g,
- $\rho_w(q)$ is the distance from q to the boundary of the channel,
- ρ_0 is the "distance of influence" of the channel boundary.

The distance of influence of the channel boundary should be taken small enough so that U_0^w is zero at q_{goal} and each intermediate goal. If this cannot be realistically[7] accomplished, the planner may be invoked to locally enlarge the channel.

In our implementation, we only consider the two cases where $\mathcal{C} = \mathbf{R}^2$ and $\mathcal{C} = \mathbf{R}^2 \times S^1$. In the first case, the distances ρ_g and ρ_w are simply the Euclidean distances from $q = (x, y)$ to $q_g = (x_g, y_g)$, and from $q = (x, y)$ to $\partial\Pi$ (the boundary of Π). In the second case, we compute the distance between $q_1 = (x_1, y_1, \theta_1)$ and $q_2 = (x_2, y_2, \theta_2)$ as:

$$d(q_1, q_2) = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2 + r\, l^2(\theta_1, \theta_2)\right]^{\frac{1}{2}}$$

where r is a scaling factor that we take equal to the maximal distance between the origin of the frame $\mathcal{F}_\mathcal{A}$ attached to $\mathcal{A}$ and the boundary of $\mathcal{A}$. The distance between two configurations is always computed within a single cell. If the cell ranges over all the orientations in S^1, we take:

$$l(\theta_1, \theta_2) = \min(|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|).$$

If it ranges over a subset of S^1, we compute $l(\theta_1, \theta_2)$ as the length of the arc connecting the orientation θ_1 to the orientation θ_2 in S^1 and contained in the angular range of the cell. We take $\rho_g(q) = d(q, q_g)$ and $\rho_w(q) = \min_{q_w \in \partial\Pi} d(q, q_w)$.

[7] If ρ_0 is very small, U_0^w increases abruptly in the vicinity of the channel boundary, which may lead to an oscillatory trajectory of the robot.

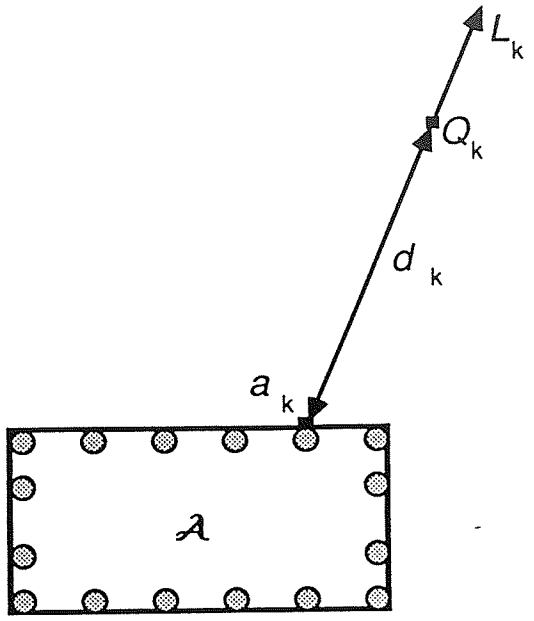

Figure 5: Contingency Detected by a Sensor

Using the control scheme described in [8], we introduce a damping term proportional to the velocity in the control of $\mathcal{A}$ and we limit the maximal speed of the robot. The damping term makes the robot decelerate when it gets close from an intermediate goal. This happens only at the final goal and at places where the channel is narrow and winding. Practically, except for short acceleration and deceleration segments, the robot navigates at maximal speed.

4.3 Contingency Potential

The workspace may contain obstacles whose shapes and locations are unknown at planning time. These unexpected obstacles can be detected by N proximity sensors equipping the robot. We denote by $S_1, ..., S_N$ these sensors. In our implementation, they are sonars mounted along the convex boundary of the robot.

We assume that, at every instant, each sensor S_k, $k = 1, ..., N$, measures the distance d_k from the point a_k in $\mathcal{A}$'s boundary, where S_k is located (see Figure 5) to an obstacle along a ray L_k fixed with respect to $\mathcal{A}$. (By convention, when S_k detects no obstacle, $d_k = \infty$.) This "perfect sensing" assumption may not be verified for a single measurement. However, the effect of a sensing error on the behavior of the robot is very brief, since another measurement will be repeated shortly after.

Let us denote by Q_k the point of $\mathcal{W}$ located at distance d_k of the boundary of $\mathcal{A}(q)$ along L_k. If $Q_k \notin \bigcup_{i=1}^{q} \mathcal{B}_i$, this point is called the **contingency** detected by S_k. At each instant, the contingencies are treated as independent point obstacles[8].

With each sensor S_k, $k \in [1, N]$, the controller associates a potential $V_k(x, y)$, which is defined *over the workspace* as follows:

[8] In order to be more rigorous, we should test that the C-obstacle corresponding to a point obstacle Q_k intersects with the channel, before calling it a contingency. Otherwise, the motion may be affected by an unexpected obstacles lying outside the channel, close to its boundary. Another, more involved improvement would be for the controller to attempt building a geometric model of the unexpected obstacles.

- If $Q_k \in \bigcup_{i=1}^{q} \mathcal{B}_i$ – i.e., Q_k is not a contingency, $\mathrm{V}_k(x,y) = 0$ for all $(x,y) \in \mathrm{R}^2$.
- Otherwise:

$$\mathrm{V}_k(x,y) = \begin{cases} \frac{1}{2} K_c (\frac{1}{\rho_k(x,y)} - \frac{1}{\rho'_0})^2 & \text{if } \rho_k(x,y) \leq \rho'_0, \\ 0 & \text{if } \rho_k(x,y) > \rho'_0. \end{cases}$$

where:

- K_c is a scaling factor,
- $\rho_k(x,y)$ is the Euclidean distance (in R^2) between the point (x,y) and Q_k,
- ρ'_0 is the "distance of influence" of a contingency.

This potential induces a force field $G_k = -\nabla \mathrm{V}_k$ over the workspace, which only applies to the point a_k. The force G_k applied at a_k is converted to a generalized force F_k as follows:

- If $\mathcal{C} = \mathrm{R}^2$, $F_k = G_k$.
- If $\mathcal{C} = \mathrm{R}^2 \times S^1$, F_k is a vector with three components. The first two components are those of G_k. The third one is the outer product[9] $O_A a_k \times G_k$, where O_A denotes the origin of the frame $\mathcal{F}_A$.

One can easily verify that $F_k(\mathbf{q}) = -\nabla \mathrm{U}_k(\mathbf{q})$, where $\mathrm{U}_k(\mathbf{q}) = \mathrm{V}_k(a_k(\mathbf{q}))$.

5 Local Minima

Although U_0 admits a single stable equilibrium state at $\mathbf{q}_{goal}$, its combination with $\sum_k \mathrm{U}_k$ may admit other local minima. A local minimum may simply be due to an unfortunate distribution of unexpected obstacles or to the complete obstruction of the channel by a relatively large unexpected obstacle. In the first case, a path may still exist in the channel. In the second case, no path exists and the planner has to be re-invoked. If all the unexpected obstacles are small and sparsely distributed, none of these two situtations are likely to happen frequently. Nevertheless, they have to be considered.

A local minimum of the potential is detected when the robot velocity gets smaller than some threshold, while the current intermediate goal is still sufficiently far away.

The controller tries to escape a local minimum by moving around the obstacle[10]. This is done by following the equipotential line or surface of the repulsive potential along the projection of the attractive force. The repulsive potential is the sum of the contingency potential and the potential produced by the boundary of the channel. This motion stops either when the trajectory is tangent to the attractive force, when the contingency potential becomes zero, or when the travelled distance is longer than some prespecified threshold. In the first case, the motion of the robot proceeds back to normal along the force induced by the total potential. In the other two cases, the robot returns to the local minimum and tries to move around the unexpected obstacle in the other direction. This is done by following the repulsive equipotential along the projection of the inverted attractive force, until one of the three conditions listed above becomes true. Again, in the first case, the motion comes back to normal. In the other two cases, the controller calls back the planner, as explained below.

[9] In order for the outer product to be non-zero, the line supporting L_k should not pass through O_A.

[10] A variety of local strategies have been proposed in [12] and [4] to deal with unexpected obstacles.

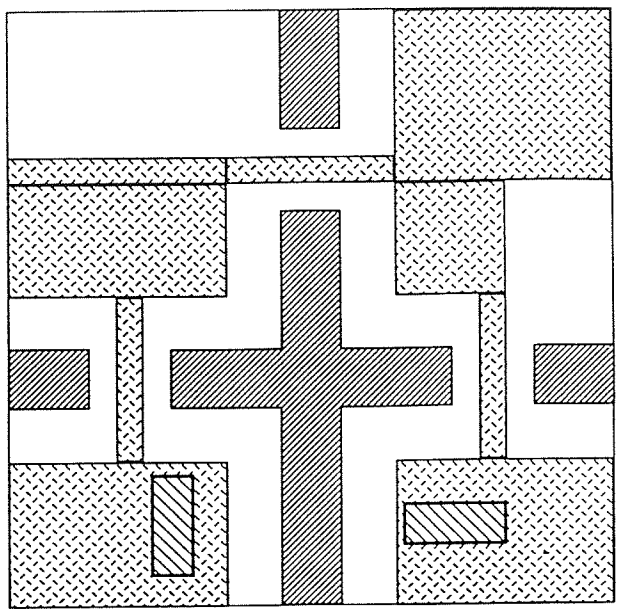

Figure 6: **Example of Channel**

Prior to getting stuck the controller kept track of all the detected contingencies Q_k and it passes them to the planner. Using this additional knowledge about the workspace and treating each contingency as a point obstacle, the planner turns all the cells of Π containing contingencies to MIXED. Then, it attempts to generate an alternative channel connecting the current configuration of the robot to $\mathbf{q}_{goal}$. In order to minimize the replanning effort, it exploits the hierarchical structure of the ccg's. It first considers the lower-level ccg containing the cell where $\mathcal{A}$'s current configuration lies and tries to find an alternative channel there. If it is not possible, the planner traverses up the hierarchy of the ccg's. If an alternative channel is ultimately found, the planner returns control to the controller with the new channel. Otherwise, the whole execution terminates with failure.

Re-using the hierarchy of ccg's not only reduces the re-planning effort. It also naturally leads the planner to produce a new channel that usually does not differ too much from the previous one. Typically, either the new channel is a subset of the previous one, or it branches back to it, hence providing a detour around the local minimum.

6 Implementation and experiments

Most of the approach described above has been implemented on an Apple Macintosh II computer. We experimented with the implemented system both with a computer simulated mobile robot and with a real robot named $\mathcal{GOFER}$. Both robots use a ring of ultrasonic range finders (sonars) to detect contingencies.

Figure 6 shows the xy projection of a channel generated by the planner for a rectangular robot that can both translate and rotate. (The origin O_A of $\mathcal{F}_A$ is at the center of the rectangle.) Figure 7 displays a path of the robot produced by following the flow of the forces generated by the channel potential in the absence of unexpected obstacle.

So far, the treatment of unexpected obstacles has only been implemented for two-dimensional configuration spaces (i.e., either

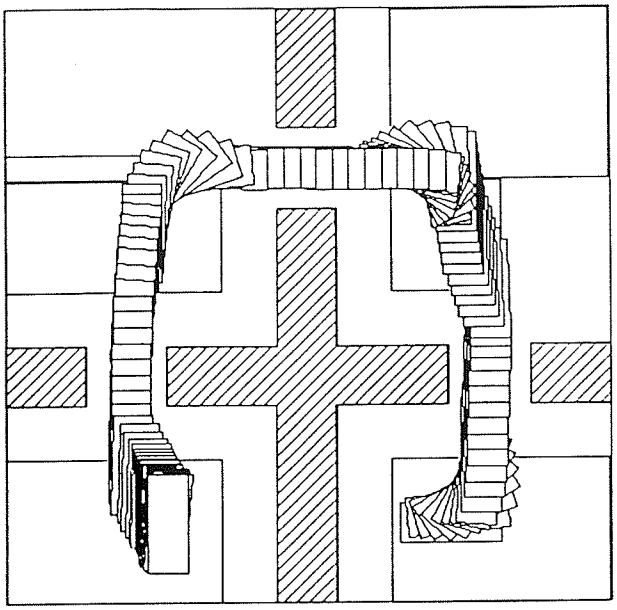

Figure 7: **Motion in a Channel**

$\mathcal{A}$ can only translate, or it is a disc) and channels extracted from a quadtree decomposition of the space. Figure 8 shows the path followed by a point robot in such a two-dimensional channel, in the presence of three unexpected obstacles. (The dark areas are the known obstacles used to construct the channel, while the lighter grey areas are the unexpected obstacles.)

7 Conclusion

In this paper, we proposed a new approach for planning and controlling the motions of a mobile robot in a partially known workspace. The approach combines a lesser-commitment planner that generates channels rather than single paths and a controller based on the potential field method. The planner shapes the channels according to the prior knowledge of the workspace. The controller makes use of the information acquired on-line about the unexpected obstacles in order to adapt the paths within the channels. The experiments conducted with the implemented system shows that this approach works well when the prior knowledge of the workspace includes the shapes and locations of the largest obstacles. The implementation remains to be completed for handling unexpected obstacles when the robot can both translate and rotate.

In the future, we plan to extend our approach to workspaces with multiple independent robots and mobile obstacles (e.g., humans). In this context, we plan to augment our framework with traffic rules in the channels aimed at making the behavior of every robot more consistent with respect to the other agents (humans and other robots).

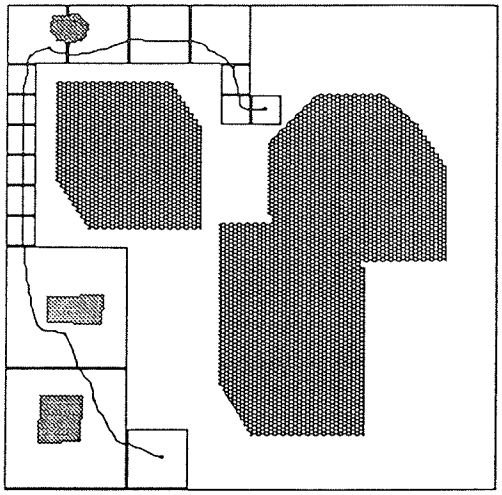

Figure 8: **Motion with Unexpected Obstacles**

References

[1] Arnold, V.I. (1978) *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York.

[2] Brooks, R.A. (1983) "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3), 190-197.

[3] Brooks, R. A. and Lozano-Perez, T. (1983) "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," *Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 799-806.

[4] Chattergy, R. (1985) "Some Heuristics for the Navigation of a Robot," *The International Journal of Robotics Research*, 4(1), 59-66.

[5] Donald, B.R. (1983) *Hypothesizing Channels Through Free-Space in Solving the Findpath Problem*, AI Memo 736, Artificial Intelligence Laboratory, MIT, Cambridge, MA.

[6] Faverjon, B. (1986) "Object Level Programming Using an Octree in the Configuration Space of a Manipulator," *IEEE International Conference on Robotics and Automation*, Atlanta, GA.

[7] Kambhampati, S. and Davis, L.S. (1986) "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation*, RA-2 (3), 135-145.

[8] Khatib, O. (1986) "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, 5(1), 1986.

[9] Koditschek, D. E. (1987) "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations," *IEEE International Conference on Robtics and Automation*, Raleigh, NC, 1-6.

[10] Latombe, J. C. (1989) "Failure Processing in a System for Designing Complex Assemblies," *Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan.

[11] Lozano-Perez, T. and Wesley, M. A. (1979) "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *Communications of the ACM*, 22(10), 560-570.

[12] Lumelsky, V. and Sepanov, A. (1986) "Dynamic Path Planning for a Mobile Automaton with Limited Information of the Environment," *IEEE Transactions on Automatic Control*, AC-31(11).

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

[14] Rimon, E. and Koditschek, D.E. (1989) "The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds," *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, 21-26.

[15] Schwartz, J.T., Sharir, M. and Hopcroft, J. (1987) *Planning, Geometry, and Complexity of Robot Motion*, Ablex, Norwood, NJ.

[16] Stallman, R. M. and Sussman, G. J. (1987) "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided CIrcuit Analysis," *Artificial Intelligence*, 9(2), 135-196.

[17] Zhu, D. J. and Latombe, J. C. (1989) *New Heuristic Algorithms for Efficient Hierarchical Path Planning*, Technical Report, Department of Computer Scienve, Stanford University, Stanford, CA.