A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty

Bertrand Bouilly, Thierry Simeon, Rachid Alami

Published in IEEE Int. Conf. on Robotics and Automation
Nagoya, Japan, May 21-27 1995

A Numerical Technique for Planning Motion Strategies of a Mobile Robot in Presence of Uncertainty

B. Bouilly, T. Siméon and R. Alami*

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex - France

Abstract

This paper addresses the problem of planning the motions of a circular mobile robot moving amidst polygonal obstacles with uncertainty in robot control and sensing. The robot is equipped with sensors which, if properly used, may provide information to overcome the uncertainty accumulated during the motions. The position sensor is based on dead-reckoning, the error then results into a cumulative uncertainty. A proximity sensor may be used to localize the robot with respect to the obstacles of the environment. The robot can also gain information by entering inside landmark areas where the position error is assumed to be bounded. We describe a planner which produces robust motion strategies composed of sensor-based motion commands which guarantee that, given an explicit model of the error accumulated by the motion commands, the robot can reach safely its goal with an error lower than a pre-specified value. It is based on a propagation of a numerical potential and on a geometric analysis of the reachability of environmental features. This planner exhibits a set of powerful capabilities: while it allows to consider motion primitives which accumulate uncertainty, it is able, whenever possible, to navigate without relocalizing the robot when the task does not impose it, and also to make a proper use of the sensors. Several examples run with the planner are presented at the end of the paper.

1 Introduction

Robot motion planning has received a widespread attention over the past decade [12]. Dealing with uncertainties constitutes an important issue of the motion planning problem since robots operating in real world settings, are faced to several sources of uncertainties arising from control errors, limited sensing accuracy and inaccurate models of the environment. Consequently, the use of motion planners which do not take explicitly into account uncertainties is limited to simple situations where the errors remain small with respect to the task tolerances. This cannot be the case neither in assembly planning nor in the context of mobile robot navigation which constitutes a challenging problem for motion planning with uncertainty.

Indeed, mobile robots are not generally equipped with an absolute localization procedure and accumulate position errors (since the error cannot be bounded, a growing of the obstacles is not sufficient). To overcome the uncertainty accumulated during the motions, the mobile robot has to be equipped with sensors that can provide additional information by identifying appropriate features of the environment. Among all the collision free paths which may connect two given configurations of the robot, only some of them may allow, at execution, to acquire enough information from the sensors and reliably guide the robot toward its goal. Dealing with uncertainty in control and sensing may therefore completely change the strategy used to plan the motions and it is preferable to *reason in advance* on the capacity of the available sensing functions, to generate a robust motion plan composed of sensor-based motion primitives.

In this paper we propose a motion planner for a circular mobile robot moving in a polygonal environment in presence of uncertainty in both sensing and control. Given an explicit model of the error accumulated by a set of motion primitives, the planner produces robust motion strategies composed of a sequence of primitives which guarantee that the robot will reliably reach its goal with an error lower than a pre-specified value. The main originality of the approach is to consider a set of motion commands which accumulate uncertainty and to plan the corrective sensor-based motions needed to reduce this uncertainty.

We addressed a similar problem in [1] for a point robot. This planner succeeded in solving a large class

*this work is supported by the CEC Esprit 3 Project 6546 PROMotion

of problems. However, it was limited to explore situations where the robot sought systematically the contact with the obstacles to localize and it was also difficult to add new sensor modalities since it involved rather complex geometric computations. The planner proposed here exhibits more powerful capabilities: it is able, whenever possible, to navigate without relocalizing the robot when the task does not impose it, and also to combine several sensor modalities.

The paper is organized as follows. In Section 2, we briefly discuss related work. Section 3 defines more precisely the problem we propose to tackle and Section 4 describes the construction of the numerical potential which is used by the planner. Finally, some simulation results are presented at the end of the paper.

2 Related Work

Motion planning with uncertainty has already attracted a lot of interest, especially in the context of automatic assembly. The reader may refer to a very interesting state of the art and discussion in [12, 13, 11] for a detailed analysis of the relevant literature. Two main approaches have been proposed:

In the *two-phase approach*, a motion plan is first generated assuming no uncertainty and then, this plan is analyzed and patched in order to finally produce a robust task plan. The analysis is mainly based on the propagation of uncertainty through the plan and its consequences on plan correctness [20, 15, 2, 18, 21, 17]. The plan is then modified by inserting complementary actions or sensor readings allowing to reduce uncertainty. The main interest of such an approach is that it can be applied to problems which can be more general than motion planning (e.g. assembly, manipulation) and to robots with a high number of degrees of freedom. However, the *a priori* decomposition of the planning process and the assumption that the plan can be locally patched, can be too restrictive for the context of mobile robot navigation.

The second approach is the *preimage backchaining approach*, originally proposed by [16] and analyzed or extended through several contributions [9, 7, 4, 12, 13]. It is based on the geometry of the Configuration Space. The underlying assumption is that it is necessary to take uncertainty explicitly into account in the planning process itself since it can have drastic consequences on the plan structure. Informally, a *preimage* of a given goal (subset of the C-space), is a set of free configurations from which a commanded motion is guaranteed to reach and terminate in the goal. Given a goal region and a start region, *preimage backchaining* consists in finding a sequence of commands such that the inverse sequence allows an iterative construction of preimages starting from the goal and resulting in a preimage containing the initial region.

While this problem, in its general formulation gives a useful computational framework, it raises "discouraging" complexity issues [4]. However this, by no means, affects its interest. It remains to find relevant instances of the general problem:

- which integrate new sensor modalities (proximity sensors, vision) and control algorithms [3, 10];
- which provide sub-classes of the problem with "better" properties [14];
- or which allow to implement algorithms of practical use even though they are not complete [1, 6].

The instance of the problem addressed in this paper and the algorithm that we propose fall in these categories. Note that we also need more elaborate models of different sensor modalities in order to compute where (in the C-Space) they can be used, and to determine the nature of the measure they can provide together with its (pre-computed) estimated uncertainty. This should of course be done, taking into account the sensor characteristics (position relative to the robot frame, range, specularity...) in a given environment [8, 19, 5].

3 Problem Statement

3.1 The robot and its environment

We consider a circular robot $\mathcal{A}$ that can translate freely in the plane among polygonal obstacles. We assume for simplicity that the environment is perfectly known. A configuration $\mathbf{q} = (x, y)$ is represented by the cartesian coordinates of the disc center and the *C-obstacles* simply correspond to the union of the generalized polygons obtained by growing the obstacles by the radius r of the robot. The environment possibly contains polygonal landmarks regions which are supposed to have a null intersection with the obstacles.

The motions of the robot are affected by control errors: when $\mathcal{A}$ is commanded to move in a given direction, the maximal deviation between the actual motion and the desired one is modelled by a *control uncertainty cone*.

3.2 Uncertainty in sensing

We consider that the robot is equipped with the following sensors:

- The position sensor is based on dead-reckoning. This technique is known to be quite sensitive to various kinds of errors and it can yield to large and cumulative errors. We model this error by an uncertain

position $\mathbf{p} = (\mathbf{q}, \epsilon(\mathbf{q}))$ corresponding to a disk centered at the sensed configuration $\mathbf{q}$ and having a radius $\epsilon(\mathbf{q}) = K.D(\mathbf{q}, \mathbf{q}_I) + \epsilon(\mathbf{q}_I)$ which linearly depends on the distance $D(\mathbf{q}, \mathbf{q}_I)$ crossed by the robot from its initial configuration $\mathbf{q}_I$. The constant K is related to the precision of the position sensor.

• The proximity sensors (or bumpers) allow to localize the robot with respect to the environment by detecting edges of the obstacles or corners made by two such adjacent edges. The information provided by such sensors is supposed to be perfect. Therefore, when a known edge is detected, the error is reset along the direction orthogonal to the edge. In this case, the uncertain position $\mathbf{p} = (\mathbf{q}, \epsilon(\mathbf{q}))$ is a segment of length $2\epsilon(\mathbf{q})$ and aligned with the edge.

• The polygonal landmarks represent regions where specific sensors (eg. vision based) can be used to detect natural or man-made features in the environment. Inside these landmark areas, the uncertainty in the measurement is modelled by a disc of fixed radius ϵ_L centered at the sensed position $\mathbf{p} = (\mathbf{q}, \epsilon_L)$.

3.3 Model of the motion primitives

The available motion primitives built upon these sensors are modelled by the type of initial situation where they can be applied, the final situation they can reach and the envelope of possible actual trajectories they can induce.

3.3.1 Motions in free-space

• $Move(x, y)$ corresponds to a nominal straight-line motion executed in position-controlled mode. Figure 1 shows the envelope of actual motions which can be executed by the robot due to error accumulation. Note that this primitive can only be used between two uncertain positions such that this envelope grown by the robot radius r completely lies in free-space.

• $Move\_to\_Wall(\theta)$ also accumulates errors but it terminates when a contact is detected by the proximity sensor (Fig 1). To avoid major localization mistake, this primitive should only be applied from a position which guarantees a correct termination onto the expected edge.

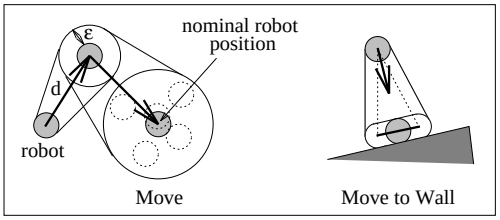

Figure 1: Error accumulated during free-space motions

• The $Move\_Landmark(x, y)$ primitive can be applied from any position $\mathbf{p} = (\mathbf{q}, \epsilon(\mathbf{q}))$ such that the disk robot grown by the uncertainty $\epsilon(\mathbf{q})$ is included in the landmark (Fig. 2). When this primitive starts, the uncertainty is reset to ϵ_L and it remains constant while the robot navigates inside the landmark.

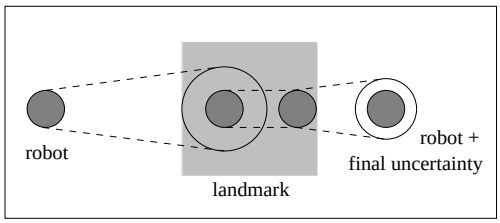

Figure 2: Localization with the $Move\_Landmark$ command

3.3.2 Motions in contact-space

• $Follow(d, [left, right])$ corresponds to a wall-following primitive which also accumulates error. The length of the segment representing the position uncertainty increases linearly with the distance d crossed during the motion (Fig. 3).

• The $Follow\_to\_Corner([left, right])$ primitive is similar but it terminates when a corner is detected. Matching the detected corner against the environment model allows to reduce the position uncertainty to zero.

• Finally, the $Switch\_Wall()$ command is used for the case of convex corners, to switch between the two edges by maintaining the contact with the corner.

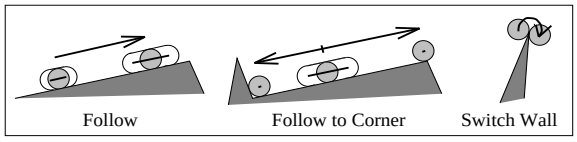

Figure 3: Localization with the wall following primitives

4 The Motion Planner

Given an initial position $\mathbf{p}_I = (\mathbf{q}_I, \epsilon_I)$ of the robot (ie. the robot can be anywhere inside the disk region of radius ϵ_I and centered at $\mathbf{q}_I$), the problem is to generate a motion plan (eg. a sequence of the available motion primitives) whose execution guarantees that the robot will not collide with the obstacles and will reach a final position $\mathbf{q}_F$ with an error less than a pre-specified value ϵ_F. The planner described below solves this problem by propagating a numerical potential over a bitmap superposed to the workspace and

by using the polygonal model of the obstacles for a geometric analysis of the reachability of obstacle edges by $Move\_to\_Wall$ primitives.

4.1 Workspace bitmap

The planner first computes a distance bitmap according to a classical "wavefront expansion algorithm" similar to the one described in [12]: The pixels lying under the edges of the obstacles are first identified and their distance is set to zero. The distance of their neighbors is then set to 1 and so on until all the free regions of the workspace have been explored. In order to decrease the approximation introduced by the L_1 metric, we alternatively consider the 1- and 2-neighborhoods at each step of the propagation. This allows a better approximation of the iso-distance circle by an octagonal shape.

4.2 Building the potential

The potential function is defined over a grid, superposed to the 2-dimensional configuration space of the robot. The potential of a given grid point corresponds to the smallest uncertainty allowing to reach this position from $\mathbf{p}_I$. The potential grid is iteratively filled, starting from the initial position, by the propagation algorithm described below. The basic structure used by the algorithm is a node structure which contains the following information:

- a grid point and the uncertainty accumulated to reach this point from the initial position,
- the distance crossed by the robot along the motion plan which produced this uncertainty,
- the parent node from which it was propagated,
- a label in the set $\{Free,\ Landmark,\ Edge,\ Vertex\}$ which indicates whether the disk robot grown by its uncertainty completely lies in free-space, in a landmark region or corresponds to a contact with an edge or a vertex of the environment obstacles.

The algorithm also maintains in a list $OPEN$, the nodes that will be possibly considered during the propagation. This list is sorted by increasing value of the node's distance and it is initialized with the node associated to the position $\mathbf{p}_I$. At each step of the propagation, the algorithm examines the 2-neighbors of the first node in the $OPEN$ list and it only retains the neighbors which satisfy the two following conditions:

- The uncertain position $\mathbf{p} = (\mathbf{q}, \epsilon(\mathbf{q}))$ (a disk or a segment) grown by the radius of the disk robot is collision-free with respect to the obstacles. The distance bitmap provides an efficient way to check this condition.
- The potential $\epsilon(\mathbf{q})$ computed for the neighbor is smaller than the potential currently stored at this position in the potential grid.

The different fields of the retained neighbors (distance, label, parent) are updated and these points are inserted in $OPEN$ according to their distance value.

The evaluation of the uncertainty associated to a given neighbor depends on both the uncertainty and the label of the current node. For example, when the node is labelled $Free$, the value of ϵ is incremented by $K.\delta$ (resp. $\sqrt{2}K.\delta$) for its 1-neighbors (resp. 2-neighbors) where δ denotes the size of the grid points. When the propagation reaches a point $\mathbf{p} = (\mathbf{q}, \epsilon)$ such that the disk of radius $r + \epsilon(\mathbf{q})$ is included in a landmark region, the uncertainty is reset to the value ϵ_L of the landmark and it remains constant for all the neighbors such that the disk of radius $r + \epsilon_L$ lies inside the landmark.

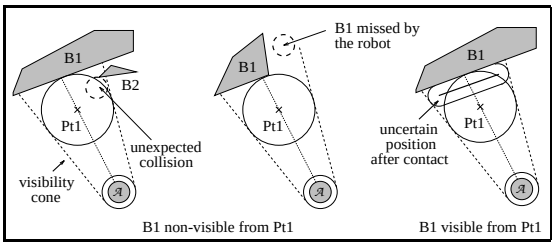

Figure 4: Edge visibility

Obstacle relocalizations are taken into account in the following way: each time the algorithm detects a neighbor that does not verify the collision-free condition because of a too large uncertainty (ie. the current node lies on the boundary of the $Free$ nodes), a visibility test is performed to determine the edges that can be reliably reached from this position by a $Move\_to\_Wall$ primitive (Fig. 4). When this test succeeds [1] for a given edge, a node is created at a position deduced from the intersection between the edge and the visibility cone associated to the $Move\_to\_Wall$ primitive. This node is labelled $Edge$ and its parent is set to the node which was currently propagated. Finally, when the propagation of such $Edge$ nodes reaches a vertex, the uncertainty is reset to zero for the corresponding neighbor. Elsewhere, the uncertainty is incremented as for the nodes having a $Free$ label.

The algorithm terminates when the goal point is reached with an admissible uncertainty (success) or when the list $OPEN$ is empty (failure).

[1] if not, the neighbor is eliminated

4.3 Computing the motion plan

When the propagation algorithm succeeds, the motion plan can be computed as follows: a "bitmap path" is first deduced by following the parents from the goal node, back to the initial one. The motion primitive used between two adjacent points along this path is determined from the labels of the nodes (cf. Table 1).

Initial label	Final label	Motion primitive
Landmark	*Landmark*	*Move_Landmark*
Free	*Edge*	*Move_to_Wall*
Edge	*Vertex*	*Follow_to_Corner*
Edge	*Edge*	*Follow*
Vertex	*Edge*	*Follow*
Vertex	*Vertex*	*Switch_Wall*
Other transitions		*Move*

Table 1: Label transitions

This solution is then transformed into a motion plan containing a reasonable number of primitives. The sequence of wall-following primitives along the same edge can be easily fused into a single motion primitive (*Follow* or *Follow_to_Corner*) followed by a *Switch_Wall* or a *Move* command. The path portions associated to free-space motions (*Move* or *Move_Landmark*) are replaced by a smaller sequence of longer straight-line motions. The algorithm used for this "smoothing" of the path guarantees that the produced motion primitives remain collision-free with the obstacles according to the model of the error accumulated by the primitives.

4.4 Remarks on the algorithm

Complexity: As previously mentioned, a given node is possibly propagated (ie. inserted in *OPEN*) only when it allows to decrease the uncertainty of the associated point in the potential grid. Therefore, in the absence of landmark or obstacle relocalization, each point is guaranteed to be propagated only once, since in this case, the uncertainty increases at each iteration of the algorithm.

Each of the l landmarks (resp. s obstacle vertices) can possibly create a local minimum of the potential when it is used for a relocalization, by reseting to ϵ_L (resp. 0) the uncertainty of a grid region (resp. point). Each relocalization can generate only once a new wave of propagation which stops when it cannot decrease anymore the uncertainty of any neighbor. In the presence of relocalizations, a given point of the grid potential can be therefore propagated in the worst case $l+s+1$ times and the algorithm is guaranteed to terminate. In practice, this number averaged over the propagated points, is much lower (see Table 2).

Properties: Each iteration of the propagation algorithm first develops the nodes having the smallest distance. Therefore, when the algorithm succeeds (ie. the propagation reaches the goal with an uncertainty lower than ϵ_F), the solution corresponds to the near-optimal motion with respect to the distance crossed by the robot along the nominal path.

In the other case, the propagation stops when the list *OPEN* is empty and all the possible localizations have been tried. Hence, if the solution was not found because the goal position was attained with a too large uncertainty, the associated goal node can be used to get the motion plan which minimizes the uncertainty.

Finally, when the problem is too constrained for connecting both initial and final positions by a robust motion plan, the regions of the potential grid attained by the propagation correspond to the workspace regions that can be reliably reached by the robot.

5 Simulation Results

The above method was implemented in C on a Silicon Graphics Indy workstation. We ran the planner on several examples with workspaces containing both polygonal obstacles/landmarks and using different types of motion strategies. The size of the bitmap grid was chosen equal to 256×256 in the examples presented in Figure 5 which illustrates some of the results obtained. For each example, the right figure shows the workspace and the nominal motion plan produced by the planner (the disk robot is displayed at the goal position). The regions reached by the propagation of the potential are displayed on the left figures which also show the robot grown by the uncertainty accumulated along the free-space portions of the motion plan. Computation times and other simulation results are given in Table 2.

Figure	1	2	3	4
Uncertainty rate	8 %	4 %	13 %	13 %
Relocalization	none	landmark	obst	obst
Total distance	29.5m	89.8m	37.5m	56.9m
Uncertainty max	2.4m	1.2m	1.7m	0.9m
Averaged (Max) propagation/point	1 (1)	2.1 (4)	1.7 (5)	1.1 (3)
CPU time	2.5s	6.5s	19.8s	12.5s

Table 2: Simulation results

Sensorless motion strategies: In the first example, the planner is given a large value for the bounded final uncertainty. The workspace contains no landmark region and the obstacle edges are too small to be reliably reached by *Move_to_Wall* commands. Hence,

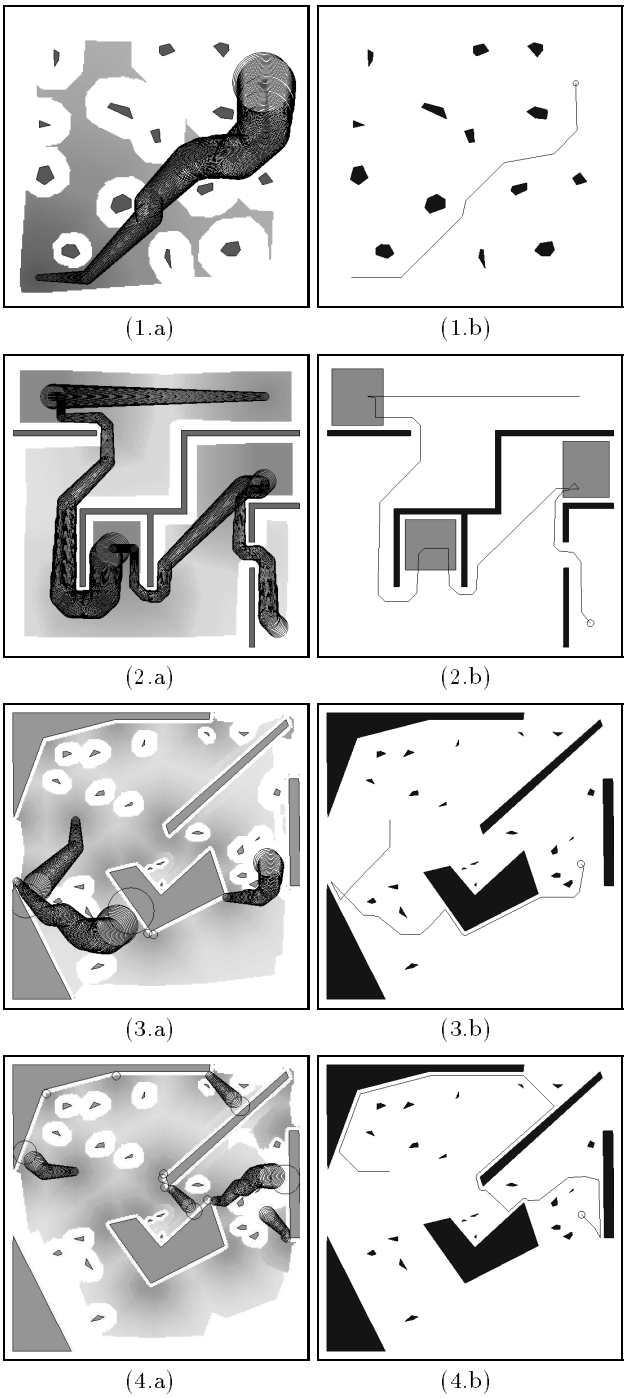

Figure 5: Paths generated by the planner

the obstacles cannot be used to localize the robot. Despite the error accumulation induced by the $Move$ primitives, the motion plan found by the planner is simply composed by a sequence of seven such primitives. This plan is guaranteed to be collision-free and it allows to reach the goal with a sufficient precision.

Influence of landmarks: The second example illustrates the influence of landmark areas (the three grey rectangles). Starting from a position in the top-right corner of the workspace, the plan should allow to reach the room of the lower-right corner with an uncertainty sufficiently small to reliably pass the small doorway. The strategy first leads the robot to cross a first landmark region and then to reach the two others (each being accessible only from the previous one), which finally makes the goal accessible. The motion primitives used to execute this path are $Move(x, y)$ to enter landmark areas and $Move\_Landmark(x, y)$ when moving inside these areas.

Influence of obstacle relocalizations: The two last examples show the effect of the relocalizations obtained by the wall following primitives in the motion plans produced by the planner. Both examples were run with the same workspace and between the same robot positions. In the first case, the strategy is aimed at minimizing the length of the nominal path. Relocalizations only occur when the uncertainty accumulated during the $Move$ commands becomes too important. The other example was run by minimizing the uncertainty instead of the distance crossed by the robot.

6 Conclusion

This paper described a motion planner for mobile robot navigation in presence of uncertainty. A numerical potential allows to propagate the error accumulated during free-space motions and also to take into account the effect of possible relocalizations by using external sensors. A geometric analysis of the reachability of environmental features determines where these sensors can be used to guarantee that they will not be confused at execution by other similar features, which may be a source of major localization mistakes. The planner has been tested on several examples and the simulation results demonstrate its ability to find rather complex motion plans in a very reasonable amount of time.

Despite its relative simplicity, the planner exhibits a set of powerful capabilities not provided by earlier similar planners: it considers sequences of motion primitives which accumulate uncertainty and it combines several sensor modalities. A nice property of the approach is also that, whenever possible, it leads to navigate without relocalizing the robot when the task does not impose it (free-space motions may be faster than wall-following and seeking for contact). Finally, another advantage of this technique compared to more complex geometrical methods may be its potential

ability to take into account more easily additional sensor modalities in a single framework. We are working on such extensions and on relaxing some assumptions currently made by the planner. In particular, it would be interesting to replace the circular model of the position error in free space by a directional uncertainty in order to precisely model the effects of the obstacle relocalizations through the motion plan. We also plan to implement the sensor-based primitives used by the planner in order to conduct experiments in real world settings and to establish the effectiveness of the approach in producing robust motion plans that can be reliably executed by a real robot.

References

[1] R. Alami and T. Siméon. Planning robust motion strategies for a mobile robot. In *IEEE ICRA'94*, San Diego (USA).

[2] R.A Brooks. Symbolic error analysis and robot planning. *Int. Journal of Robotics Research*, 1(4), 1982.

[3] R. Brost. Dynamic analysis of planar manipulation tasks. In *IEEE ICRA'92*, Nice (France).

[4] J. Canny. On computability of fine motion plans. In *IEEE ICRA'89*, Scottsdale (USA).

[5] I. Collin, D. Meizel, N. Le Fort and G. Govaert. Local map design and task function planning for mobile robots. In *IROS'94*, Munich (Germany).

[6] J. Najera, F. De la Rosa and C. Laugier. Planning robot motion strategies under geometric constraints. In *IROS'94*, Munich (Germany).

[7] B.R. Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence*, 37:223–271, 1988.

[8] B.R. Donald, J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. TR 91-1248, Cornell University, Dec 1991.

[9] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *Int. Journal for Robotics Research*, 5(1):19–45, Spring 1986.

[10] A. Fox, S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans (parts I & II) In *IEEE ICRA'93*, Atlanta.

[11] K. Goldberg, M. Mason, A. Requicha. Geometric uncertainty in motion planning: Summary report and bibliography. IRIS TR 297, USC Los Angeles, 1992.

[12] J.-C. Latombe. Robot Motion Planning. *Kluwer Academic Publishers*, 1991.

[13] J-C. Latombe, A. Lazanas, S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence*, 52(1):1–47, Nov 1991.

[14] A. Lazanas, J-C. Latombe. Landmark-based robot navigation. TR STAN-CS-92-1428, Stanford University, May 1992 (to appear in *Algorithmica*).

[15] T. Lozano-Perez. The design of a mechanical assembly system. A.I Memo 397, MIT Artificial Intelligence Lab., MIT, Cambridge, Dec 1976.

[16] T. Lozano-Perez, M.T. Mason, R.H. Taylor. Automatic synthesis of fine motion strategies for robots. *Int. Journal of Robotics Research*, 3(1), 1984.

[17] I. Mazon, R. Alami. Representation and propagation of positioning uncertainties through manipulation robot programs.. In *IEEE ICRA'89* , Scottsdale.

[18] J. Troccaz, P. Puget. Dealing with uncertainty in robot planning using program proving techniques. *The Fourt ISRR*, Santa Cruz (USA), March 1987.

[19] H. Takeda, J-C. Latombe. Sensory uncertainty field for robot navigation. In *IEEE ICRA'92*, Nice.

[20] R.H. Taylor. Synthesis of manipulator control programs from task-level specifications. AIM 228, AI Laboratory, Stanford, July 1976.

[21] R.H. Taylor, V.T. Rajan. The Efficient Computation of Uncertainty Spaces for Sensor-based Robot Planning. In *IEEE IROS'88*, Tokyo (Japan)