FP6-IST-002020

COGNIRON

The Cognitive Robot Companion

Integrated Project

Information Society Technologies Priority

D3.5.1
Architecture of the placement planner for human-robot close interaction

Due date of deliverable: 31/12/2004
Actual submission date: 31/12/2004

Start date of project: January 1st, 2004 **Duration:** 48 months

Organisation name of lead contractor for this deliverable:
LAAS-CNRS

Revision: Final
Dissemination Level: PU

Executive Summary

In this document, we present ongoing work on a planner that is specifically created for Human-Robot Interaction problems in LAAS/CNRS Laboratory under the European project Cogniron. The presence of humans in the environment raises new issues in the classic motion-manipulation planning where social rules and safety measures must be taken into account.

We claim that such a planner can be used not only to plan safe robot paths, but also to plan good, socially acceptable and legible paths. Indeed, besides models of the robot and the environment, we intend to build a planner that takes explicitly into account the human partner by reasoning about his accessibility, his vision field and potential shared motions.

The work that we have accomplished in this first phase involve the refinement of the models and constraints to be dealt with and the overall architecture of the planner. Indeed, such a planner will have to take into account not only geometric and kinematic constraints but also sensor models and symbolic and qualitative constraints. The second phase will focus on algorithmic aspects.

Role of Motion and Manipulation Planning in Cogniron

A robot companion such it is studied in the Cogniron project will have close interactions with humans. This will be demonstrated in Key Experiments (see next section).

Relation to the Key Experiments

The work developed here will conduct to the development of concepts and algorithms that will be implemented and illustrated in the Key Experiments: a so-called placement and manipulation planner that will allow to plan socially acceptable robot motions in the vicinity of humans in navigation and in manipulation contexts. Its use will be illustrated in Key Experiment 1 (navigation) and Key Experiment 2 (navigation and manipulation).

1 Concepts of a Planner for Human-Robot Interaction Problems

1.1 Introduction

In this very moment, lots of robots are working to build computers, cars, planes, are orbiting around earth or exploring the space. Although their jobs are quite different from one another, their working environment have a common aspect: no humans. None of the autonomous robots today is really "living" in human environment and interacting meaningfully with humans. Generally and also naturally, the robot existence is avoided nearby humans for security reasons and also for they are not intelligent enough to simplify humans everyday life. This lack of intelligence comes from the fact that interacting with humans, even co-existing with humans in an environment without an explicit interaction, needs to take into account safety and "social" issues.

The interaction between humans and robots can be in several different types: verbal interaction, visual interaction and physical interaction. In verbal interaction, we use the dialogue capabilities of the robot to communicate with a human, for example a robot receiving orders verbally. Visual interaction imposes visibility constraints to the robot for viewing the human and being viewed by the human. An example for such situations is the case where robot must look at a certain direction and also must be seen by the human. And the physical interaction consists of exchanging objects between a robot and a human and planning the motion of the robot in presence of humans. Obviously we cannot draw exact boundaries between these different types by the fact that every one of them has relations with others. For example, visibility is a defining constraint for motion planning and also motion and manipulation have serious effects on the field of view. The work described here is mainly focused on motion and manipulation planning and also we will discuss sensor-based constraints and extensions to take into account symbolic/qualitative constraints in the same planning process.

To plan feasible and safe motions for robots, we need to have a minimum set of capabilities that the robot should possess:

- be able (see deliverable 7.1.1) to acquire a model of its surrounding environment (Cogniron functions CF-NAV, CF-OR)
- be aware of the human presence and intention (CF-PTA, CF-TBP, CF-ACT);
- be able to receive orders from humans and evaluate them (CF-DLG, CF-ROR) .

Our target here is to provide material for Cogniron functions CF-NHP (Navigation in the presence of humans) and CF-MHP (Manipulation in the presence of humans).

In this document, we will go through the literature related to recent work on planning in human environments and then present the framework that we propose, based on recent results obtained in sensor-based as well as in intricate manipulation and symbolic problems.

1.2 State of the Art

Although several authors propose motion planning or reactive schemes in human contexts, there is no contribution that tackles globally the problem as we propose to do. A key issue for this kind

of problems where humans exist nearby robots is safety. In industrial robotics safety is assured by not allowing humans in a certain perimeter around robots and having emergency stop buttons [11]. Actually there is no interaction in these cases. So when we carry the robot next to a human, we must be very clear about safety criteria.

In a recent work made by Nonaka et al. [13], the concept of safety has been studied by two aspects: ”physical” safety and ”mental” safety of humans. Physical safety means that the robot does not physically injure humans. Mental safety, on the other hand, means that the motions of the robot do not cause any un-comfort or inconvenience like fear, shock, surprise to humans. Unlike physical safety, mental safety constraints are related to the reactions given by humans to robot motion. To obtain these reactions, Nonaka et al. uses a virtual reality device mounted to a human displaying a virtual robot. The choice of the robot's shape must be carefully chosen for a human to correctly predict its motion. In their experience they used a humanoid robot. The subject sees the robot sitting in front of him and moving only his arms, his torso and his head. As the virtual robot moves the subject's reactions are stored into four type of emotion: surprise, fear, disgust and unpleasantness.

An interesting result appeared by the fact that the human reaction depends not only on the velocity of the motion but also on the motion order of the robot's parts. For example, when robot arm and body move at the same time we have a very low level of disgust and unpleasantness but when body begins his movement before the arm, these emotions take higher levels. And also making the motions slow, rather than fast, lessen unpleasant emotions.

The user studies that UH conducts in the framework of Cogniron, based on scenarios that are close to what we intend to implement will certainly give us valuable information.

On the other hand, physical safety is the absolute need of the human-robot interaction. This can be assured at the hardware and software design process of the robot. The solid structure of the robot can be made softer using light and soft material that minimizes the consequences of an impact. Ikuta et ale. classifies the safety strategies into two different types: control strategies and design strategies[10]. Control strategies include controlling the distance, speed, moment of inertia and stiffness. Design strategies include the material constraints as weight, cover, surface and shape. They propose a general method for evaluating safety that takes into account not only the physical structure of the robot but also a control architecture that minimizes the danger. This minimization is made by a danger-index that contains the maximum impact force, momentum and approaching velocity. With these informations, they were able to create a space-time danger chart and a simulator that helps to evaluate overall danger, optimize safety control and understand the danger of every direction at all times and calculate a safe path for robot.

Although the danger-index criterion gives us a safe path, it does so by considering only the end effector point trajectory with respect to the human. Another work in this direction is made by Kulic and Croft where a robot moves in front of a human by minimizing a danger criterion [12]. The danger criterion is calculated by the robot inertia and the relative distance between the robot and a human. Two types of danger criterion are defined: sum-based and product-based. The sum-based criterion is a function that can be interpreted as a quadratic attractive function attracting each link of the robot towards its base. The relative distance can be modeled as a repulsive force between the human and the robot center of mass. The product-based criterion, on the other hand, uses the inertia to calculate this function.

The distance between the robot and the human is calculated by using a set of enveloping spheres that describes the objects in the environment. Initially a small number of large spheres are used for each

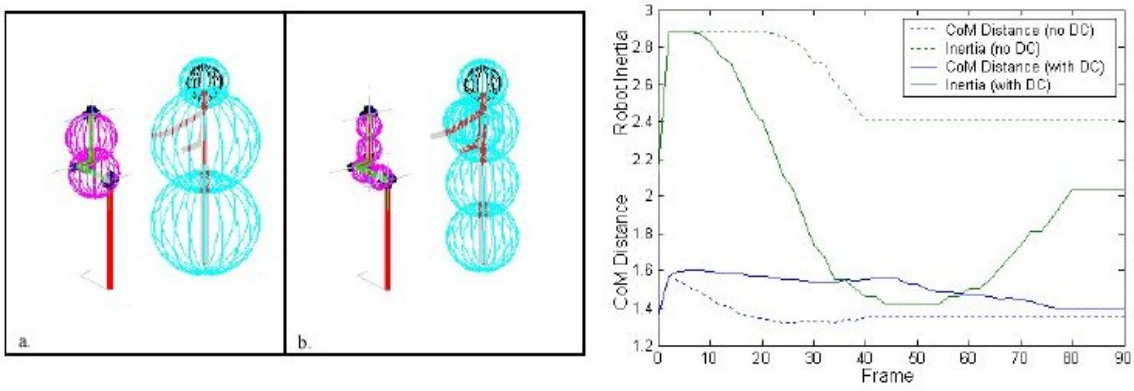

Figure 1: a) a non-interactive task b) an interactive task c) The results of utilization of Danger Criterion

object. In case of collision, the spheres are decomposed into smaller spheres. This process goes on until a non intersecting set of spheres is found and the distance between them is returned or until the number of spheres exceeds a threshold. Figure 1-a,b shows enveloping spheres in case of a non-interactive task where the robot and the human are covered entirely and an interactive task where the robot should give an object to a person, so the person's hand is excluded from spheres. As we can see in Figure 1-c, the results of using a danger criterion improves greatly the safety although the model has been made for a very simple 2D form of the robot and the human.

With these approaches physical safety is assured by avoiding collision with humans and by minimizing the intensity of the impact in case of a collision. Another direction towards human-robot interaction is the research made for smart wheelchairs. Although there is not a real interaction between the chair and the human in a direct sense, the motion of the chair taking into account the human's comfort is an interesting issue. We can see a certain aspect of interaction in the work made by Rao et al. where an "intelligent" wheelchair interacting with the sitting human by using a interface and planning chair's motion [15]. Their work is mainly focused to the interface aspect but they implemented a number of obstacle avoidance control algorithms that actually work in a real wheelchair.

In the literature we can also find some work trying to imitate human motions for having a better understanding of how humans behave in social environments. A recent work made by Althaus et al. [2] makes robot behaving like humans in a conversation. We must note that this behavior "imitates" humans self-placement in a conversation. The expected robot behavior is:

1. The robots enters a room and looks around for people.
2. If any humans are present, the platform approaches them.
3. The robot stands next to the person(s) and initiates or joins a discussion.
4. When the discussion is over or the robot is asked to do something else, it leaves the person(s) towards exit of the room.
5. The system resumes some old plan, responds to some new request, or wanders around in its environment.

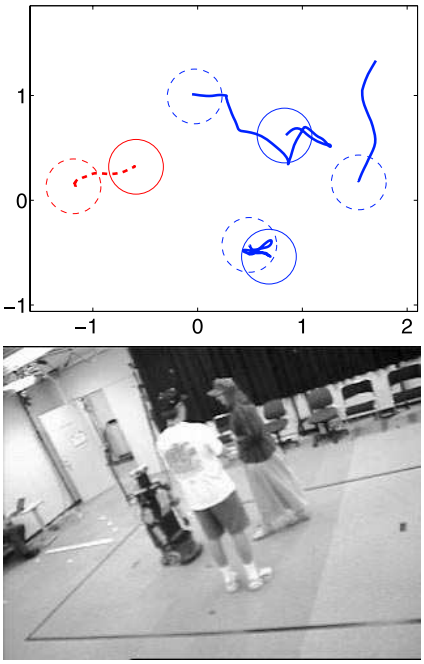

Figure 2: The person on the right hand side is leaving the formation

The navigation and recognition of possible conversation places are made by topological map and a state diagram that contain the map of the environment in forms of corridors, doors, rooms where the motion of the robot can be in two states: approach and join. So the labeling of the topological map allows to have the map and the plan in the same structure. Approaching to a group of humans is controlled by the distance to the group and the direction of the group's members to align robot's orientation. The speed of approach is inversely proportional to the distance for not to scare people.

To maintain the formation with humans, the robot scans continuously with his sonar to calculate the average distance between humans in the environment. In case of a newcomer or a leaving person (Figure 2), the average distance changes and the robot aligns himself. If a person steps closer towards the robot, its contribution gets bigger and the gaze of the robots turns towards him. After a pre-defined time, the robot "leaves the conversation". The work here in fact does focus not the planning process but on a distance control mechanism. Safety in this method is assured by setting a threshold to the distance between the robot and humans.

Safety and confort: Another approach that deals not only with safety but also implicitly comfort issues is the work on velocity profiles along a planned trajectory [1], where a robot calculates its speed and its trajectory to optimize the execution time and also to guarantee that no collision will occur. What is of interest here is that the planner takes into account explicitly the sensor capabilities (field of view, occlusion) and the dynamics of the environment. Movable obstacles (humans) have a known maximum speed but can be hidden.

By incorporating robot and humans dynamics as well as sensory constraints the planner can adopt a pro-active strategy. By pro-active we mean that the robot is always in a state of expectation regarding the possibility of a mobile object impinging onto its path from regions invisible to its sensor. This pro-active state is reflected in the velocity profile of the robot, which guarantees in the worst case

scenario, the robot, from its side would not collide with any of the moving objects that can interfere with its path

Since the sensors have a certain range, it's likely necessary to slow down in some places of the robot trajectory where the sensors field of view is blocked by narrow passages or corners. In figure 3-a, we see that, in order to be sure not to collide with moving objects thats pop out from the door, the robot must decelerate until the sensors' range covers the invisible area.

As a result of this, we see that the shortest path do not always give us the fastest path due to the velocity constraints. Instead of minimizing path length, this method minimizes the time by constructing an optimal velocity profile (Figure 3-b). With this approach, the robot passes the corners or narrow passages with a wider angle so that he has a better view. Note If we consider the robot as a wheelchair, this path will be more comfortable for the human sitting on top thanks to the speed that adapts to the field of view.

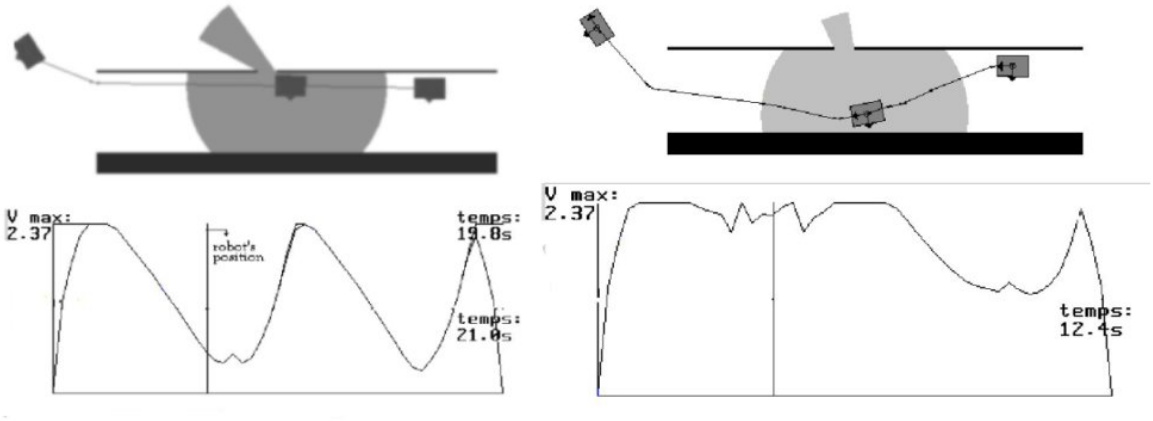

Figure 3: a) Planned robot's path and robot's velocity profile b)Modification of the path for a better velocity profile

Motion coordination: Multi-robot planning techniques might be applicable to our problem. Indeed, the robot and the humans can be seen as several "robots" that have to be coordinated. Consequently, we have also studied a bunch of multi-robot planning methods. An effective coordination method proposed by Berg and Overmars described in [18] makes use of state-time grids to achieve a coordination between multiple robots. In this approach, a grid (called state-time grid) containing the time periods of occupancy of each point of the roadmap is constructed in order to find a collision-free path in state-time domain. The obstacles in the environment are pre-calculated at the creation stage of the roadmaps and an A* search is performed to find the shortest path efficiently in state-time grids. Another multi-robot path coordination method is the use of scheduling algorithms [17] and optimization algorithms [14] to coordinate a set of robots whose trajectories are already calculated. This method gives good results and can be applied to a high number of robots.

For the conclusion we see that the distance, the inertia and the velocity are sufficient quantities that we can use in our planner to assure safety. But although the safety issues are covered, we can't see in literature a planning process that takes into account explicitly the other constraints that we have proposed: social acceptance, intention legibility.

1.3 A framework for hybrid task planning

In this section we will present the framework LAAS is developing for multi-robot task planning and its current instance, called “aSyMov” and discuss how it can serve as an inspiration for elaboration a planner for human-robot interactive tasks.

Although the domain independent planner improved dramatically, their use in robotics is very limited due to the gap between the representation of the real world and the real world itself. On the other hand, path planners are dedicated to geometric problems. State of the art motion planning systems can even plan for manipulation tasks: they handle complex geometric constraints and for instance they can compute the intermediate Pick&Place actions which are sometimes needed for re-grasping objects. However such planners cannot handle symbolic relations in a generic way. aSyMov has been specially designed to address robot planning problems where geometric constraints cannot be simply abstracted in a way that has no influence on the obtained plan.

aSyMov [8] [4] (a Symbolic Move3d) is an original planner which uses the competence of a task planner (MetricFF [9]) and a path and manipulation planner (Move3d [16]) in a hybrid way. At each step of the planning process both symbolic and geometric data are considered.

1.3.1 Geometric reasoning issues

In order to solve complex manipulation problems that may induce re-grasping for instance, the geometric part of aSyMov uses a multi-roadmap approach [7]. For Three different kinds of roadmaps are used: “transit” roadmaps that embed collision-free motion of the robot, “transfer” roadmaps that contains collision free nodes for the motion of the robot holding an object (we consider it as a new robot composed by the empty one and the object) and so-called “$Grasp \cap Placement$” roadmaps. The search for nodes that “link” different types of roadmaps allow to “capture”, in a systematic way, the topology of a manipulation planning problem in a given environment and then, to search for possible sequence of actions (motion, grasp, release).

In the Cogniron context, the presence of the human in the robot environment gives us new motion criteria that must be taken into account. One of these concepts is the ”visibility”. As the word implies, the visibility gives us two constraints about the robot vision and also the human’s vision (field of view).

When considering robot navigation tasks, the robot must be fully of partially visible to the human for the human comfort and understanding. The person near the robot must see or guess the robot motions to feel comfortable but also humans must not have to do an extra effort to see the robot. Consequently, and depending on the context, we must carefully plan robot paths to be fully or partially visible to an identified human by avoiding invisible points.

Another class of tasks for which we can define a visibility criterion is manipulation tasks in which we need to have a high resolution model of the manipulated object. Since the sensor accuracy increases proportionally to the distance, the robot must focus its ”gaze” to the object to achieve fine motions. In a scenario where a robot gives an object to the human, the latter visibility constraint serves not only to allow the robot to achieve better motions but also helps humans to have a better understanding of robot intention.

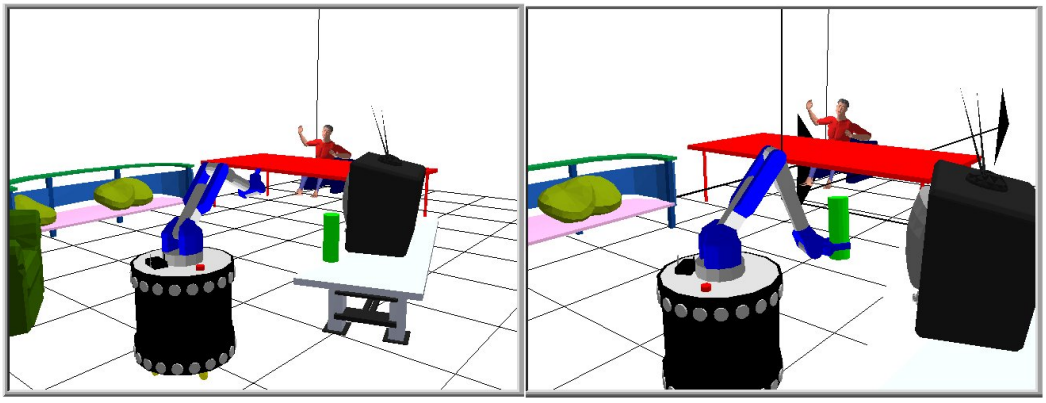

Figure 4: An example of Human-Robot interaction where the robot brings the object.

Visibility should also be taken into account when gaze-contact or dialogue actions have to take place.

By the word ”visibility”, we mean that at some stages of the plan (or all along the plan), the robot (or part of it) must be visible to the human. Fine motions might also be made under a continuous complete visibility. In order to realize these constraints, we will use a multi-roadmap approach similarly to aSyMov. As this approach allows us some level of modularity, visibility notion can be implemented as a different type of roadmap. Although it can contain various criteria (for example the comfort obtained by velocity control [1], the physical structure of the robot or the motion of the robot), the comfort issue that we would like to discuss can be derived by selecting suitable human-robot configurations where the human will feel comfortable if a manipulation task occurs.

In figure 4 we can see a scenario where a robot gives an object to a human. We can think of many ways for the robot to hand the object. The exchange of objects may occur anywhere within the human reach (we suppose here that the human is sitting and not willing to stand up). In figure 6, we see that some positions are not comfortable for a human (because of fear&safety issues or because they require an extra effort for the human). Figure 7 illustrates two suitable configurations for grasping the object are shown. We believe that this comfort criterion can be also represented in roadmaps. We may even use relative roadmaps for humans (or robots) by pre-calculating an accessibility graph covering the human. Having a pre-calculated graph will help us to store a number of information associated to the nodes such as the level of “comfort” for a configuration (or a zones). Similar zones are heavily used in Characters Animation. For instance, we have developed [5] a computation method by spherical shell methods using a Random Loop Generator algorithm. Figure 5 illustrates such computation.

Figure 5: a) Accessibility zones for a human and a robot. b) the object must be at the intersection of the two zones to be manipulated c) Bigger objects need to be taken into account

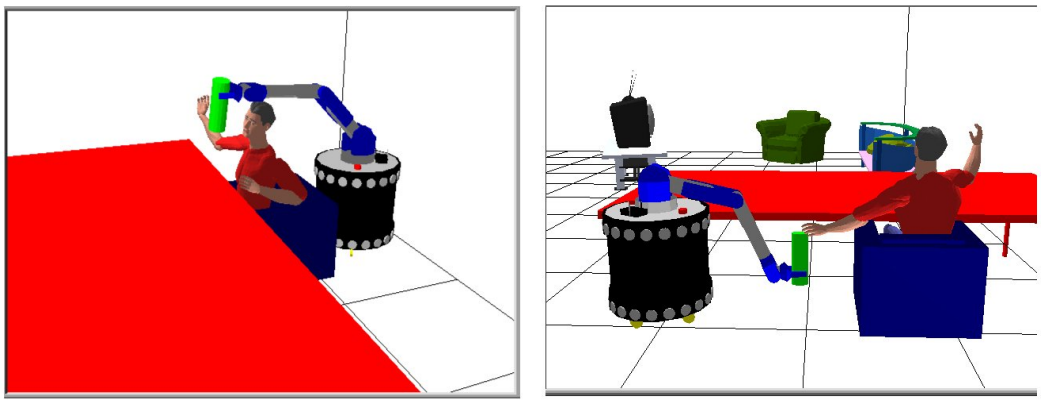

Figure 6: Two uncomfortable ways to give the object to the human

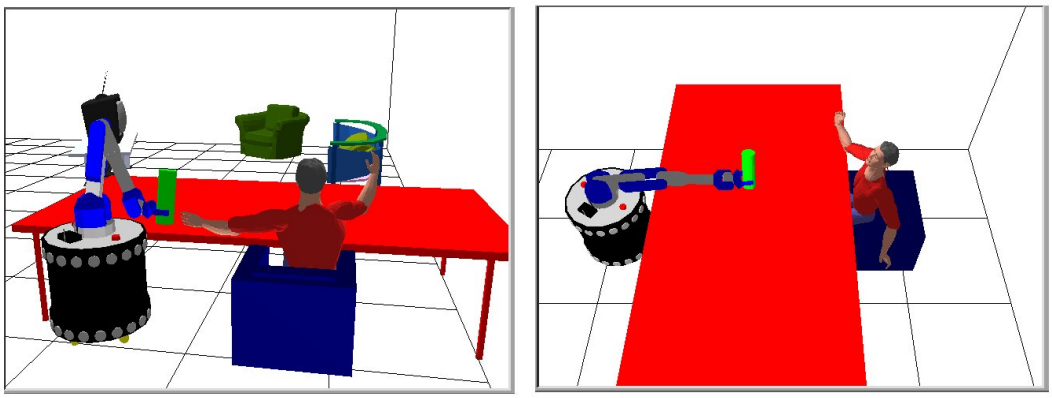

Figure 7: Two comfortable ways to give the object to the human

1.3.2 Symbolic reasoning issues

As already mentioned, we essentially focus here on two types of robot tasks for Human-Robot Interaction problems:

1. Navigation in presence of humans which involve robot motion taking account safety, comfort, visibility, etc issues.
2. On the other hand manipulation task that involve a detailed reasoning on humans accessibility as well as rules and protocols that will influence the robot paths and behaviors.

To give an example of such constraints, we can consider a ”gaze contact” action. At some stage of the plan, the robot must not only look to the human but also be sure that the human is looking at it. In order to provide this, a gaze contact action will either move the robot to a place of interest for the human or make a motion to draw the attention of the human.

In aSyMov tasks and actions are allocated to the robot by the symbolic planning level. The locations of robots and objects are represented as “symbolic positions” that correspond to a set of configurations that satisfy a given property. For instance, the configurations where a robot `R` can grasp an object `O` will be denoted by `P_R_TI_TA_O`. Indeed such a symbolic position corresponds to a transition between a transit motion (`TI`) and a transfer motion (`TA`). As a consequence of the grasp action, a new robot `R-O` will be created. It may place (transition between `TA` and `TI`) the object at a position denoted by `P_R-O_TA_TI`

In order to tackle properly the interaction between the symbolic and the geometric aspects of robot problems, a framework is defined where geometric consequences of symbolic actions can be expressed. We use three main types of symbolic parameters: robot, movable object (a particular type of robot) and position. The following predicates are defined. Note that PDDL2.1[6] is chosen to represent this concepts.

- **(composed ?r1 ?r2 ?r3)**: the composition of two robots ?r1 and ?r2 is possible and the result is a third robot ?r3. (eg: `(COMPOSED R O R-O)`)
- **(belongs-to ?p ?r ?roadmap-type)**: a position ?p belongs to a roadmap of type ?roadmap-type for a robot ?r; examples of roadmap-types are `TI` (transit) and `TA` (transfer). (eg: `(BELONGS-TO P_R_TI_TA_O R TI)`)
- **(has-purpose ?p ?pos-type)**: is used to declare that a position ?p is dedicated to a special treatment. For example, initial and goal positions play a special role and are often associated to a unique geometric position. This predicate is also used to specify areas in which a robot must be located to apply a purely symbolic action.
- **(connection ?p1 ?p2)**: denotes that it is possible to find a connection between two positions ?p and ?p2 which do not belong to the same roadmap. (eg: `(CONNECTION P_R_TI_TA_O P_R-O_TA_TI)`)
- **(on ?r ?p)**: robot ?r is situated at the symbolic position ?p.

Using this set of predicates, we can specify actions which add or remove "*on*" predicates. Three main actions are defined: *goto* (motion), *grasp* (composition) and *ungrasp* (decomposition). Note that these actions are not built-in the planner but are simply specified thanks to the predicates described above. Here is the grasp action:

```
(:action grasp
 :parameters (?r    - robot           ?p1 - position
              ?o    - (either obj robot) ?p2 - position
              ?newrobot - robot         ?p3 - position)
 :precondition (and (on ?r ?p1)
                    (on ?o ?p2)
                    (belongs-to ?p3 ?newrobot TA)
                    (composed-robot ?newr ?r ?o)
                    (connection ?p1 ?p3)
                    (connection ?p2 ?p3))
 :effect (and   (not (on ?r ?p1))
                (not (on ?o ?p2))
                (on ?newrobot ?p3)))
```

"Purely" symbolic predicates can be added and associated to actions that may have no effect at the geometric level. For instance, a predicate *"have-magnetic-key"* can be used as a precondition for an *open-door* action.

Currently we are experimenting the usefulness of this representation for Human-Robot interaction (HRI) situations. A state of the world must be defined in order to provide all the necessary information at any step of the planning process. This state must include all the elements in the environment along with their capabilities at that moment. In the HRI scenarios that we consider, the state of the world will be essentially composed of physical state of the robot, the human (posture) and the objects (geometry, labels). In first step, we are experimenting PDDL2.1. However, we may have to create our own representation less general but more suitable for HRI.

1.3.3 A Hybrid planning architecture

One crucial aspect here that potentially allows to tackle realistic problems is the control of the potential complexity of the underlying geometric problem, by a task-oriented symbolic planner.

The connection between the levels relies on selection of suitable roadmaps as well as the estimation of the costs associated to paths, and to the "switch" between roadmaps. The overall planning process performs a search for suitable actions and incremental instantiation of symbolic positions.

The control of the search can be implemented using aSyMov "update or extend" algorithm. The planner can start with a set of empty roadmaps that it will explore incrementally during its search. Such roadmaps can be re-used for subsequent planner calls. At each step it will have to choose between trying to find a plan with the level of knowledge it already has acquired (the roadmaps as they are), or to "invest" more in a deeper knowledge of the topology of the different configuration spaces that it manipulates (selection and expansion of some roadmaps).

Note that an action that has not been validated previously may become valid after a roadmap expansion. Consequently, the front search must also attach to each state the list of applicable but non-currently valid geometric actions. This list must be re-considered whenever the roadmaps have been updated by a roadmap expansion. So we cannot remove simply a state from the front search. We stop the search after a predefined number of step of the core procedure which may add a new state to the front search: the *extend state* algorithm (fig 8).

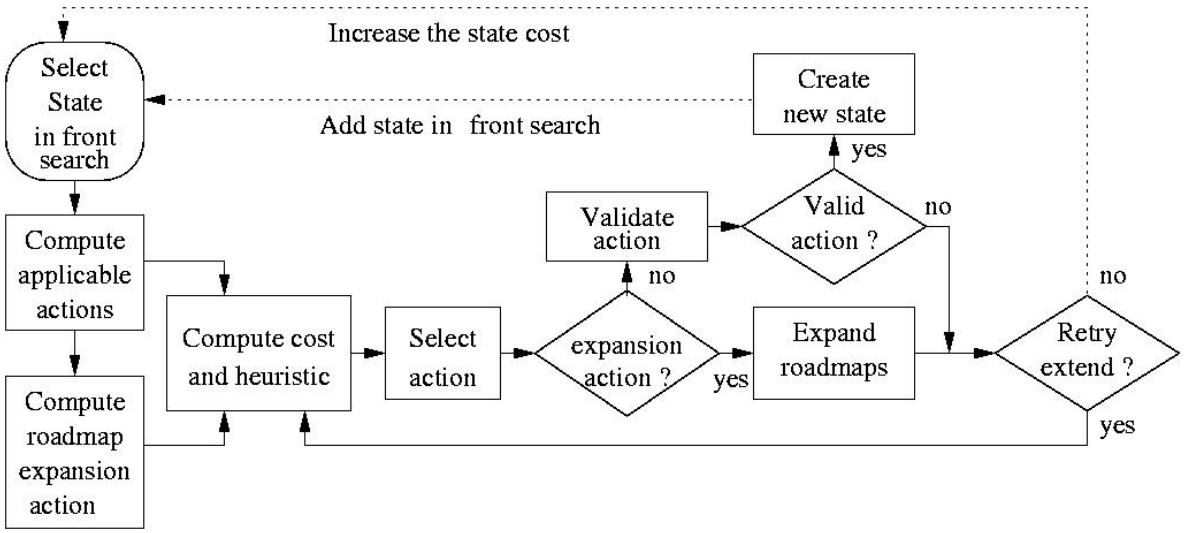

Figure 8: Algorithm to extend a state from the front search

The first operation is to compute the *applicable* actions on the basis of the symbolic part of the state. Thus, we ensure the satisfaction of the symbolic constraints. We introduce complementary actions called *roadmap-expansion* which implement an incremental roadmap expansion algorithm.

The second operation computes costs and heuristics of the actions. The selection includes a random part. Even if the best action (i.e. for a symbolic planner) is often the most likely, all feasible actions have at least a small probability to be chosen. If the symbolic planner is complete and the roadmap expansion has the probabilistic completeness property, our planner keeps this probabilistic completeness property. The non-existence of solution is so unprovable.

If the selected action is a *roadmap-expansion*, the planner selects and expands a set of roadmaps which may have interests to find a plan (interests partially computes on the basis of the relaxed solution). If the selected action is an *applicable* one and if it has geometric consequences (action which contains “on” predicates in its effects), the planner tries to validate it.

At the end of the *extend state* procedure, the planner changes the cost of some state of the front search. If the procedure fails to extend the state, its cost is increased.

When the goal is reached a set of post-processing steps not detailed here can be performed in order to clean the plan (redundant actions can appear), to detect actions that can be performed in parallel, to smooth the robot trajectories and to synchronize multi-robot motions.

This approach is very suitable for HRI problems where we had to chose whether exploring the roadmap or passing to the next action. The ”compute cost” part of the algorithm may serve us to

specify various criteria to take into account all along the planning process.

2 Future Work

In this deliverable a general scheme of a planner that deals with Human-Robot Interaction problems is defined. It is based on a multi-roadmap approach. A high-level symbolic/qualitative representation of the tasks allows to conduct a hybrid search process.

We are creating a set of basic tasks to study the capabilities of a PDDL-like representation and to see if that language will be powerful enough to express the notions we would like to implement.

A model of human and robot will be designed carefully according to our needs and to results obtained by the other project partners. For instance, the user studies that are conducted in the framework of Cogniron, based on scenarios that are close to what we intend to implement will certainly give us valuable information. We are using our Move3d platform to implement the future planner. A number of optimizations are foreseen for the geometric part of our planner as simplification from 3D to 2D in appropriate situations, entropy guided path planning [3] or visibility planning [19]. Algorithms will be defined in 2D for navigation and 3D (with detailed robot and human kinematic chains) for manipulation.

3 References

3.1 Reference documents

Paper “Safe Pro-active Plans and their Execution”, K. Madhava Krishna, R. Alami and T. Simeon, (submitted to “IEEE Robotics and Autonomous Systems”)

References

[1] R. Alami, T. Simeon, and K. Madhava Krishna. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. *Proc. in IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[2] Philipp Althaus, Hiroshi Ishiguro, Takayuki Kanda Takahiro Miyashita, and Henrik I. Christensen. Navigation for human-robot interaction tasks. *Proc. in IEEE Int. Conf. on Robotics & Automation, New Orleans, LA*, 2004.

[3] Brendan Burns and Oliver Brock. Single-query entropy-guided path planning. *accepted paper to ICRA*, 2005.

[4] S. Cambon, F. Gravot, and R. Alami. A robot task planner thet merges symbolic and geometric reasoning. *European conference on artificial intelligence*, 2004.

[5] C. Esteves, Gustavo Arechavaleta, and Jean-Paul Laumond. Planning cooperative motions for animated characters. *In the proceedings of the Intenational Symposium on Robotics and Automation*, 2004.

[6] M. Fox and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Technical Report, University of Durham, UK*, 2001.

[7] F. Gravot, R. Alami, and T. Simeon. Playing with several roadmaps to solve manipulation problems. *IEEE International Conference on Intelligent Robot & Systems*, 2002.

[8] F. Gravot, S. Cambon, and R. Alami. asymov: a planner that deals with intricate symbolic and geometric problems. *International Symposium of Robotics Research (ISRR)*, 2003.

[9] Hoffmann and Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence ResearchIOS Press Vol. 14*, pages 253–302, 2001.

[10] Koji Ikuta, Hideki Ishii, and Makoto Nokata. Safety evaluation methods of design and control for human-care robots. *The International Journal of Robotics Research, Vol 22, No. 5, pp. 281-297*, 2003.

[11] D. Kulic and E. Croft. Strategies for safety in human robot interaction. *Proc. in IEEE Int. Conf. on Advanced Robotics, pp. 810-815*, 2003.

[12] D. Kulic and E. Croft. Safe planning for human-robot interaction. *Proc. in IEEE Int. Conf. on Robotics & Automation, New Orleans, LA*, 2004.

[13] Seri Nonaka, Kenji Inoue, Tatsuo Arai, and Yasushi Mae. Evaluation of human sense of security for coexisting robots using virtual reality. *Proc. in IEEE Int. Conf. on Robotics & Automation, New Orleans, LA*, 2004.

[14] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *Algorithmic Foundations of Robotics V pages 221-237, Springer-Verlag, Heidelberg, Germany*, 2003.

[15] R. S. Rao, K. Conn, S. H. Jung J. Katupitiya, T. Kientz, V. Kumar, J. Ostrowski, S. Patel, and C. J. Taylor. Human-robot interaction: Application to smart wheelchairs. *Proc. in IEEE Int. Conf. on Robotics & Automation, Washington, DC*, 2002.

[16] T. Simeon, J-P Laumond, and F. Lamiraux. Move3d: a generic platform for path planning. *4th International Symposium on Assembly and Task Planning*, 2001.

[17] T. Simeon, S. Leroy, and J. Laumond. Path coordination for multiple mobile robots: a resolution complete algorithm. *IEEE Transactions on Robotics and Automation, volume 18*, 2002.

[18] Jur P. van den Berg and M. Overmars. Roadmap-based motion planning in dynamic environments. *Technical Report*, 2004.

[19] Pengpeng Wang and Kamal Gupta. View planning via maximal c-space entropy reduction. *Fifth International Workshop on Algorithmic Foundations of Robotics, Nice, France*, 2002.

Safe Proactive Plans and their Execution (submitted to "IEEE Robotics and Autonomous Systems"

K. Madhava Krishna[a] R. Alami[b] T. Simeon[b]

a: International Institute of Information Technology, Hyderabad - India
b: LAAS-CNRS, 31077 Toulouse Cedex - France

Abstract: *Presented in this paper a methodology for computing the maximum velocity profile over a trajectory planned for a mobile robot. Environment and robot dynamics as well as the constraints of the robot sensors determine the profile. The planned profile is indicative of maximum speeds that can be possessed by the robot along its path without colliding with any of the mobile objects that could intercept its future trajectory. The mobile objects could be arbitrary in number and the only information available regarding them is their maximum possible velocity. The velocity profile also enables to deform planned trajectories for better trajectory time. The methodology has been adopted for holonomic and non-holonomic motion planners. An extension of the approach to an online real-time scheme that modifies and adapts the path as well as velocities to changes in the environment such that both safety and execution time are not compromised is also presented for the holonomic case. Simulation and experimental results vindicate the efficacy of this methodology.*

1 Introduction

Several strategies exist for planning collision free paths in an environment whose model is known [9]. However during execution, parameters such as robot and environment dynamics, sensory capacities need to be incorporated for safe navigation. This is especially so if the robot navigates in an area where there are other mobile objects such as humans. For example in figure 1, the robot would require to slow down as it approaches the doorway, in anticipation of mobile objects to emerge from there, even if it does not intend to make a turn through the doorway.

A possible means to tackle the above problem at the execution stage is to navigate the robot at very small speeds permanently. In-fact reactive schemes such as the nearness diagram approach [11] operate the robot at minimal velocities through out the navigation. However incorporating the computation of velocity profile at the planning stage would circumvent not only the problem of permanently conservative velocities throughout navigation but also for a modification of the trajectory towards better time lengths such as in figure 2

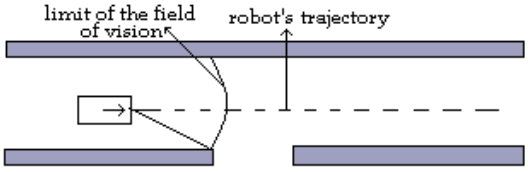

Figure 1: It entails that a safe robot slow down while approaching the doorway

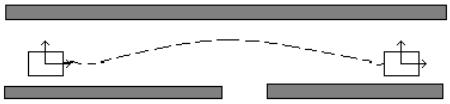

Figure 2: a longer path could be faster due to better velocities

Presented in this paper a novel pro-active strategy that incorporates robot and environment dynamics as well as sensory constraints onto a collision free motion plan. By pro-active we mean that the robot is always in a state of expectation regarding the possibility of a mobile object impinging onto its path from regions invisible to its sensor. This pro-active state is reflected in the velocity profile of the robot, which guarantees in the worst case scenario, the robot, from its side would not collide with any of the moving objects that can interfere with its path. In fact the ability of the algorithm to a-priori compute velocities for the entire trajectory accounting for moving objects moving in whatsoever direction is the essential novelty of this ef-

fort. A similar kind of strategy for the aforementioned objective does not appear to have been confronted in robotic literature so far.

As is always the case, planned paths and profiles need constant modification at the execution stage due to changes in environment. For example a profile and path that was planned for an environment with a closed doorway needs to be modified during real-time if the doorway is found open. Also addressed in this article the problem portrayed in figure 3. Given an initial trajectory planned for a particular environment how does the robot modify its trajectory while new objects (not necessarily interesecting the robot's trajectory) are introduced into the environment such that the basic philosophy of ensuring safety as well as reducing time-lengths of the path continue to be respected. Simulation and experimental results are presented to indicate the efficacy of the scheme.In [1] we had reported how the maximum velocity profiles can be computed for any generic planner and in [8] we presented initial simulation and experimental results of the reactive version of [1].

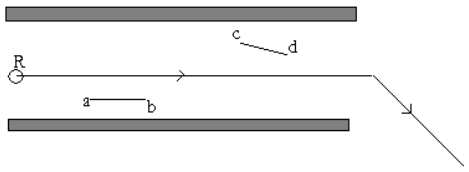

Figure 3: How does the robot adapt its path in the presence of new segments (a, b) and (c, d) respecting the philosophy

Related work can be cited in the areas of modifying global plans using sensory data obtained during execution for overcoming uncertainty accumulated during motions [3] and those that try to bridge the gap between planning and uncertainty [10] or planning and control [7], [2]. The velocity obstacle concept[13][5] bears resemblance to the current endeavor in that they involve selection of a robot velocity that avoids any number of moving objects. The difference is that in the present approach the only information about the mobile object available is the bound on their velocity. The direction of motion and their actual velocities are not known during computation of the velocity profile. The work of Strachniss [14] also involves considering robot's pose and velocities at the planning phase. A path is determined in the(x, y) space and a subgoal is chosen. A sequence of linear and angular velocities, (v, w), is furnished till the subgoal is reached. It however does not speak of reducing the time-length of path by modifying it and the dynamics of the environment does not seem to affect the computation of the velocity profile.In [12] a policy search appoach is presented that projects a low dimensional intermediate plan to a higher dimensional space where the orientation and velocity are included. As a result better motion plans are generated that enable better execution of the plan by the robot. The current effort has similar ties to [12] at the planning level but also extends it to a suitable reactive level in the presence of new obstacles encountered at execution

2 Problem Definition

The following problems are addressed in the paper, given:

- A robot $\mathcal{R}$ modelled as a disc and equipped with an omnidirectional sensor having a limited range R_{vis}. We call $\mathcal{C}_{vis}$ the visibility circle, centered at robot's position with radius R_{vis}. The paths of $\mathcal{R}$ are sequences of straight segments or straight segments connected with circular arcs of radius ρ in case of a non-holonomic robot. The robot's motion is subject to dynamic constraints simply modelled by a bounded linear velocity $v \in [0, v_{rm}]$ and a bounded acceleration $a \in [-a_{-m}, a_m]$. The maximum possible deceleration a_{-m} need not equal the maximum acceleration a_m.
- A workspace cluttered by static polygonal obstacles $\mathcal{O}_i$. The static obstacles can hide possible mobile objects whose motions are not predictable; the only information is their bounded velocity v_{ob}.

Problem 1: Given a robot's path $\tau(s)$ computed by a standard planner [9], determine the maximal velocity profile $v_\tau(s)$ such that, considering the constraints imposed by its dynamics, the robot can stop before collision occurs with any of the mobiles that could emerge from regions blind for the robot at position $s \in \tau(s)$. For example the velocity profile dictates that the robot in figure 1 slow down near the doorway in expectation of mobile objects from the other side. We call $MP = (\tau(s), v_\tau(s))$ a **robust motion plan**. The velocity profile allows us to define the time $T(\tau)$ required for the robust execution of path τ:

$$T(\tau) = \int_0^L \frac{ds}{v_\tau(s)}$$

Problem 2: Modify the planned trajectory such that the overall trajectory time $T(\tau)$ is reduced. For

example the path of figure 2 is traversed in a shorter time though longer than figure 1

Problem 3 Adapt the path and velocities reactively in the presence of new objects not a part of the original workspace such that the criteria of safe velocities and reduced time-lengths of paths continue to be respected. This is portrayed in figure 3

3 From path to robust motion plan

The procedure for computing the maximum velocity profile $v_\tau(s)$ delineated in sections 3.1, 3.2 and 3.3 addresses the first problem. The constraints imposed by the environment on the robot's velocity is due to two categories of mobile objects. The first category consists of mobiles that could appear from anywhere outside the boundary of the visibility circle C_{vis}. The second category involves mobiles that could emerge from shadows created in C_{vis} due to stationary objects.

3.1 Velocity constraints due to the environment

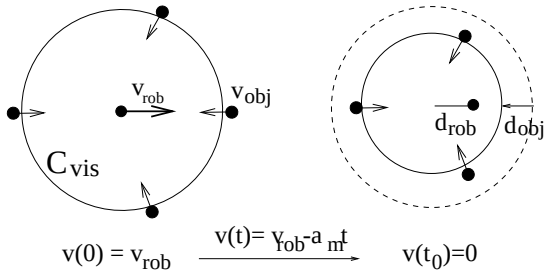

Figure 4: Mobile objects may appear anywhere onto the C_{vis}'s contour.

No obstacles in $\mathcal{C}_{vis}$ In the simple case where the robot's position is such that no static obstacle lies inside $\mathcal{C}_{vis}$, a moving object may appear (at time $t = 0$) anywhere onto $\mathcal{C}_{vis}$'s boundary (Fig. 4). Let V_{rb} denote the maximum possible robot velocity due to a mobile at the boundary. At time $t_0 = v_{rb}/a_{-m}$ (ie. when the robot is stopped), the distance crossed by the object is $d_{obj}(v_{rb}) \leq v_{ob}v_{rb}/a_{-m}$. Avoiding any potential collision imposes that $R_{vis} \geq d_{rb}(v_{rb}) + d_{obj}(v_{rb})$, where $d_{rb} = v_{rb}^2/2a_{-m}$. The condition relates v_{rb} to the sensor's range R_{vis} as:

$$v_{rb} = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}R_{vis}} \qquad (1)$$

Influence of shadowing corners Static obstacles lying inside $\mathcal{C}_{vis}$ may create shadows (eg. see the grey region of Figure 5) succeptible to contain mobile objects. The worst-case situation occurs when the mobile remains unseen until it arrives at the *shadowing corner* of a polygonal obstacle. Since the mobile's motion direction is not known it is best modeled for a worst case scenario as an expanding circular wave of radius $v_{ob}t$ centerd at (d, θ)

$$(X(t) - d\cos\theta)^2 + (Y(t) - d\sin\theta)^2 = v_{ob}^2 t^2$$

Let us first consider that the robot's path τ is a straight segment. Considering that the intersections between the circular wave and the robot's segment path, should never reach the robot before it stops at time $t_0 = v_{rs}/a_{-m}$ yield to the following velocity constraint:

$$v_{rsv}^4 - 4(a_{-m}d\cos\theta + v_{ob}^2)v_{rsv}^2 + 4a_{-m}^2 d^2 \geq 0 \qquad (2)$$

Here v_{rsv} is the maximum possible robot velocity due to the shadowing vertex under consideration. The solution of eq. 2 gives v_{rsv}, as a function of (d, θ).

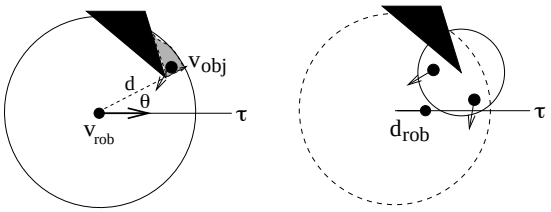

Figure 5: Mobile objects may also appear in the shadows of static obstacles

This solution only exists under the condition $v_{ob} > \sqrt{a_{-m}d(1-\cos\theta)}$, ie. when the object's velocity v_{ob} is sufficiently high to interfere with the robot's halting path. Otherwise, the shadowing corner does not constrain the robot's velocity which can be set to v_{rm} the maximum bound on robot's velocity.

A similar reasonning can be applied to the case where the robot traverses a circular arc path of radius ρ. This case however leads to a non linear equation that needs to be solved numerically to derive the maximal velocity [4]. The expression that needs to be solved for computing the maximum velocity at a given point on a circular arc is of the form

$$\begin{aligned}((v_{rsv}^2 v_{ob}^2)/a_{-m}^2) + 2\rho^2\cos(v_{ob}^2/2a_{-m}\rho) + \\ 2d\rho\sin((v_{ob}^2/2a_{-m}\rho) - \theta) \\ = d^2 + 2\rho^2 - 2d\rho\sin\theta \end{aligned} \qquad (3)$$

3.2 Computing the shadowing corners

The problem of determining the set of shadowing corners needed for the velocity computation in 3.1 is the problem of extracting those vertices of the polygonal obstacle to which a ray emitted from the robot's center is tangential (Figure6). The set of shadowing corners can be easily extracted from an algorithm that outputs the visibility polygon [15] as a sorted list of vertices.

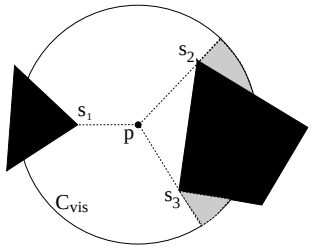

Figure 6: Shadowing corners: among the three vertices of $\mathcal{V}(p)$, only s_2 and s_3 create shadows (the line going through s_1 is not tangent to the left obstacle

3.3 Computation of maximum velocity profile $v_\tau(s)$

While the methodology for computing the maximum velocity profile delineated here is essentially for a holonomous path, its extension to the non-holonomous case is not complex.

1. A holonomic path τ, consisting of a sequence of straight line segments ab, bc, cd (fig 7) is deformed into a sequence of straight lines and clothoids to ensure continuity of velocities at the bends [6]. The maximum deviation from an endpoind to its clothoidal arc (depicted as e in figure 7) is dependent on the nearest distance to an object from the endpoint under consideration

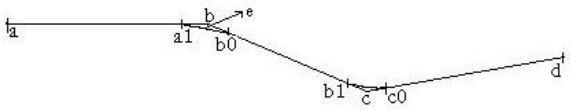

Figure 7: A holonomic path deformed into a sequence of straight segments and clothoidal arcs

2. The linear velocity along a clothoid is a constant and the maximum possible linear velocity considering robot dynamics alone is calculated for each of the clothoidal arc $a1b0, b1c0$ according to [6] and are represented as $v_c(a1), v_c(b1)$.
3. The straight segment $aa1$ is discretized into M equally spaced points, excluding the endpoints of the segment, viz a and $a1$. We denote the first such point as a_1 and the last of such points as a_M. The point of entry into the clothoid, viz. $a1$ is also denoted as a_{M+1}.
4. For each of the N points, a_i, the steps $4a$ to $4e$ are repeated.

4a Maximum possible velocity that a robot could have such that it can come to a halt before colliding with objects that enter into the robot's field of vision from the boundary is computed as $v_{rb}(a_i)$ according to equation 1

4b Velocity of the robot due to stationary obstacles inside the robot's field of vision that create shadows is computed as $v_{rsv}(a_i)$ according to equation 2. The minimum of all the velocities due to such vertices is found and denoted as $v_{rs}(a_i)$.

4c The maximum possible velocity of the robot at a_i due to environment is then computed as

$$v_{re}(a_i) = min(v_{rb}(a_i), v_{rs}(a_i)) \tag{4}$$

4d Velocity of the robot at a_i due to its own dynamics is given by

$$v_{rd}(a_i) = \sqrt{v_r^2(a_{i-1}) + 2a_m s(a_i, a_{i-1})} \tag{5}$$

The above equation is computed if $v_{re}(a_i) > v_r(a_{i-1})$. Here $s(a_i, a_{i-1})$ represents the distance between the points a_i and a_{i-1} and a_m represents the maximum acceleration of the robot.

4e The eventual velocity at a_i is given by

$$v_r(a_i) = min(v_{rd}(a_i), v_{re}(a_i), v_{rm}) \tag{6}$$

Here v_{rm} represents the maximum robot velocity permissible due to servo motor constants.

5. The velocity at the endpoint $a1$ is computed as $v_r(a1) = min(v_r(a1), v_c(a1))$ and this would be the linear velocity with which the robot would traverse the clothoid.
6. Steps $6a$ and $6b$ are performed by going backwards on each of the N points from a_N to a_1

6a If $v_r(a_i) > v_r(a_{i+1})$ then modified maximum possible velocity at a_i is computed as

$$v_{rd}(a_i) = \sqrt{v_r^2(a_{i+1}) + 2a_{-m} s(a_i, a_{i+1})} \tag{7}$$

6b Finally the maximum safe velocity at a_i is given as $v_r(a_i) = min(v_r(a_i), v_{rd}(a_i))$

7. Repeat steps 3 to 6 for all the remaining straight segments to obtain the maximal velocity profile over a given trajectory τ as $v_\tau(s) = \{v_r(a), v_r(a_1), ..., v_r(a_N), v_r(a1), v_r(b_1), ..., v_r(d)\}$

3.4 Modifying planned trajectory for better time lengths

The knowledge of the maximum velocity profile over a trajectory is utilized to tackle the problem posed in section 2 of reducing the overall trajectory time of the path. The procedure for reducing trajectory time at the planning stage involves random deformation of the planned path and evaluating time along this path. The modified path becomes the new trajectory if time along it is lesser than along the original trajectory. The process is continued till over a finite number of attempts no further minimization of trajectory time was possible. Prior to delineating the algorithm it is to be noted that the set of all collision free space of the workspace is denoted as C_{free} and the current tracteory of the robot as $\tau_c(s)$. A point of discretization on a trajectory discretized into N parts is denoted as $p(s_i), i \in \{1, 2, ..., N\}$. The corresponding configuration of the robot at those points is denoted by $q(s_i)$. The algorithm is given as Algorithm 1

Step 8 of the algorithm is carried out by searching for a collision free configuration which would displace the path away from the shadowing vertex responsible for the lowest velocity at s_i . Step 11 adapts the displaced path as the new current path if its trajectory time is lesser than the current path. $N_{attempts}$, is the number of unsuccessful attempts at minimizing trajectory time before the algorithm halts.

3.5 Memorization of Sensor Information

The computation of the velocity profile at a given point on the robot's trajectory incorporates the robot's field of vision at that point. This field can change appreciably between two successive instances of computation. For example in figure 8 the robot at position a has full field of vision of the corridor that is transverse to the robot's trajectory. However at position b the robot is blind to the zone shown in darker shade of gray. Hence it needs to slow down its velocity as it moves further down to c since it envisages the possibility of a moving object approaching it from the corners of the stationary objects. These corners are the starting areas of the robot's blind zone at b.

However if the robot could memorize the scenario cognized earlier it can retain this memorized image for computing its velocity profile during execution of the planned path. In such a case if the robot had not seen any moving objects in close proximity at a it can make use of this information at b to have a velocity profile from b that is greater than the one computed in the absence of such memorization. Fig 8 shows the zone memorized by the robot in darker shade. The

Algorithm 1 Globally reducing trajectory time

1: $N_{try} \leftarrow 0$
2: **while** $N_{try} < N_{attempts}$ **do**
3: Discretize current trajectory $\tau_c(s)$ into N_p where N_p is selected based on minimum discretization distance between two points.
4: Set $flag \leftarrow 0$
5: **for** $i = 1$ to N_p **do**
6: Compute least velocity at s_i due to shadowing vertices as $v_{rmin}(s_i)$
7: **if** $v_{rmin}(s_i) < v_{rm}$ **then**
8: Find a configuration $q(s_p) \in C_{free}$ and $s_p \notin \tau_c(s_k), k \in \{1, ..., N_p\}$ such that $q(s_p)$ is reachable from $q(s_i)$.
9: Find a point s_r on the remaining part of the trajectory, i.e, $s_r \in \tau_c(s_j); i < j \leq N_p$ such that $q(s_r)$ is reachable from $q(s_p)$.
10: Form a new trajectory through s_i, s_p, s_q and denote it as $\tau_n(s)$
11: **if** $T(\tau_n) < T(\tau_c)$ **then**
12: discretize τ_n into N_q points.
13: $\tau_c \leftarrow \tau_n$
14: $N_p \leftarrow N_q$
15: Set $flag \leftarrow 1$
16: **end if**
17: **end if**
18: **end for**
19: **if** $flag = 0$ **then**
20: $N_{try} \leftarrow N_{try} + 1$
21: **end if**
22: **end while**

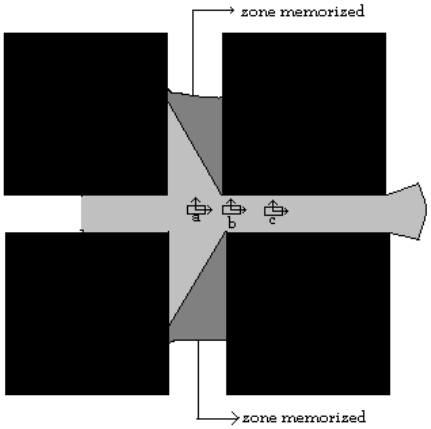

Figure 8: Memorization of previous scenes

contour of the memorized area represents the blind-zone of the robot at b, from where mobile objects can

emanate. The area in lighter shade of gray is the visibility polygon for the robot at b. With the passage of time the frontier of the memorized area shrinks due to the advancement of the imagined mobiles from the initial frontier. The details of the memorization scheme are given below.

Memorization is fruitful when a vertex hitherto non-shadowing begins to cast a shadow thereby hiding regions which were previously visible. The set of all vertices that are currently visible, shadowing and were at some prior instant visible, non-shadowing is denoted by $Vsns$. For every vertex $ve \in Vsns$ a corresponding vertex is associated and called the blind vertex. The blind vertices are of three categories explained through figure 9 where the vertex a, non-shadowing for the robot at a becomes shadowing when the robot is at q. Correspondingly the vertex c of the triangular obstacle which was visible and shadowing when the robot was at p had become invisible when the robot moved to q. Simultaneously one of the other end-points of b, viz. a, would also become inevitably invisible at q. Vertices of the kind b fall in second category. If b was already outside C_{vis} at p the intersection of C_{vis} with the segment ab, namely o is identified as the third category of blind vertex. The set of all such vertices is denoted by Vbs. These vertices are advanced by a distance $v_{ob}\Delta t$ where Δt is the time taken by the robot between p and q to new virtual locations along the line that connects those vertices to a. At q the velocity is computed due to the closest of the vertices in the set Vbs at their virtual locations instead of a, which is otherwise the vertex for which equation(2) is computed.. Such a trend continues till the distance between the robot to the closest hypothetical vertex is lesser than the actual distance of the robot to a.

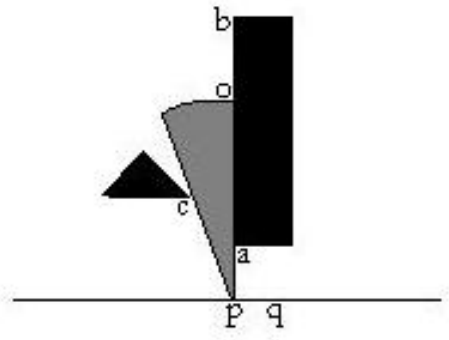

Figure 9: Three categories of blind vertices

The memorization part of the algorithm is given in algorithm2. The set of all visible shadowing vertices is denoted by Vsh

Algorithm 2 Memorization effects on velocity

```
for each vertex v_e ∈ Vsh do
  if v_e ∈ Vsns then
    for each vertex v_b ∈ Vbs associated with v_e do
      Advance v_b by v_ob Δt
    end for
    Denote the distance from the robot's current location, s_c, to the closest of all advanced vertices, v_bc as d_cvb
    if d(s_c, v_e) < d_cvb then
      Compute velocity due to the virtual vertex v_bc through equation 2
    else
      Compute velocity due to the actual vertex v_e through equation 2
    end if
  end if
end for
```

4 From Plan to Execution

The velocity profile, $v_\tau(s)$, is a sequence of maximum velocities calculated at discretized locations along the trajectory $\tau(s)$. The computation of the velocity profile at the execution stage is not at the same locations where the profile was computed at planning due to odometric and motor constraints. Moreover if there are changes in the environment it entails modifying the trajectory and hence the velocities. During execution it is computationally expensive to compute the profile for the entire remaining trajectory, hence the profile is computed for the next finite distance, given by, $d_{safe} = d_{max} + nd_{samp}$, where $d_{max} = v_{rm}^2/(2 * a_{-m})$, represents the distance required by the robot to come to a halt while it moves with the maximum permissible velocity afforded by motor constants. And $d_{samp} = v_{rm}t_{samp}$ is the maximum possible distance that the robot can move between two successive samples (time instants) of transmitting motion commands, where time between two samples is t_{samp}.

The main issue here is what should be the distance over which the velocity profile needs to be computed during execution such that it is safe. A velocity command is **not** considered safe if it is lesser than the current velocity and **not** attainable within the next sample. The velocity is constrained by environment as well as robot's own dynamics and hence the role

of either in creating such an unattainable velocity is probed below.

Effect of Environment Mobile objects that can emerge from corners in a head-on direction cause the greatest change in velocity over two samples. Figure 10 shows one such situation, where the rectangular object casts a shadow and is susceptible to hide mobiles. Let the current velocity of the robot at a due to the object be v_1. Let the velocity at a distance, s, from a, at b (fig 10) due to the object be v_2.

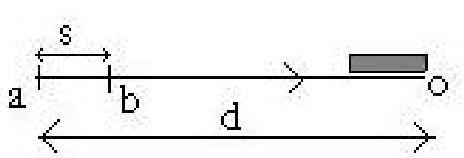

Figure 10: The effect of the rectangular object that could hide possible mobiles on robot's velocity at locations a and b

The velocities at a and b are given by

$$v_1(a) = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}d} \tag{8}$$

$$v_2(b) = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}(d-s)} \tag{9}$$

Hence

$$v_1^2(a) - v_2^2(b) = 2a_{-m}s + 2v_{ob}(\sqrt{v_{ob}^2 + 2a_{-m}(d-s)} - \sqrt{v_{ob}^2 + 2a_{-m}d}) \tag{10}$$

Evidently the second term on the right hand side of equation 10 is negative, since the second square root term is less positive than the first. Hence $v_1^2(a) - v_2^2(b) \leq 2a_{-m}s$. Therefore the velocity at b, v_2 can be attained from the velocity at a, v_1 under maximum deceleration, d_m, irrespective of the maximum velocity of the mobile or the robot's own motor constraints. This was intutively expected since the robot's velocity at any location is the maximum possible velocity that guarantees immobility before collision, its velocity at a subsequent location permitted by the environment would be greater than or equal to the velocity at the same location obtained under maximum deceleration from the previous location. In other words for safeness of velocity going purely by environmental considerations it would suffice to calculate the velocity, for the next sampling distance alone, for without loss of generality, $d = d_{samp}$.

Effect of robot's dynamics The robot needs to respect the velocity constraints imposed while nearing the clothoidal arcs and eventually while coming to the target. The robot can reach zero velocity from its maximum velocity over a distance of d_{max}, computed before. Hence $d_{max} + d_{samp}$ represents the safe distance over which the velocities need to be computed.

4.1 Online path adaptation for better trajectory time

The third of the problems outlined in section2 is tackled herewith. During navigation the robot in general comes across objects hitherto not a part of the map. The robot reacts to these new objects in line with the basic philosophy of safe as well as time reduced paths. The adaptation proceeds by finding locations over a finite portion of the future trajectory where drops in velocity occur and pushing the trajectory away from those vertices of the objects that caused these drops to areas in free space where higher velocities are possible. A search is made through the newly found locations of higher velocities for a time reduced path.

Generalized Procedure The generalized procedure for adapting the path in presence of new objects is delineated through figure 11

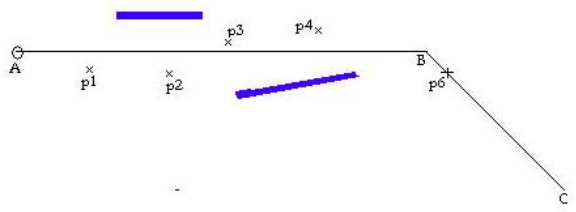

Figure 11: A trajectory in presence of new objects. The points marked with crosses represent locations through which a path is searched for better time-length

1. On the trajectory segment that is currently traversed, AB in figure 11, enumerate the vertices of objects that reduce the velocity of the robot.
2. The positions are found on AB where the influence of vertices is likely to be maximal.
3. These positions are pushed by distances $d_p = k(v_l - v_r)$, where v_l and v_r are the velocities at that location on the path due to the most influential vertices on the left and right of the path. These new locations are denoted as $p1, p2, p3, p4$

(fig 11) and maintained as a list provided the velocity at the new locations is more when compared with the original ones. $p6$ is the farthest point on the robot's trajectory visible from its current location at A

4. On this set of locations $A, p1, p2, p3, p4, p5, p6$ starting from the current location at A, find a trajectory sequence shorter in time than the current sequence of $A, B, p6$ if it exists.
5. The steps 1 to 4 are repeated until the robot reaches the target.

It should be noted that

- If a collision with an object is detected, a collision free location is first found that connects the current location with another location on the original trajectory and this new collision free path is further adapted for a time-reduced path if it exists.
- While the velocities are computed over a distance d_{safe}, that part of the remaining trajectory that is visible from the current location is considered for adapting to a better time-length.

5 Analysis and Results at the Planning Stage

In this section the results of incorporating the velocity profile computation as a consequence of considering robot and environment dynamics and sensor capacities at the planning stage and the subsequent adaptation of paths to better time-length is analyzed. Figure 12 shows the path computed by a typical holonomous planner [9] and its corresponding velocity profile. The velocity corresponding to the robot's location on the trajectory (shown as a small circle) is marked by a straight line labeled m on the profile. The dark star-shaped polygon centered at the robot depicts the visibility of the robot at that instant and is called the visibility polygon. The figure is a snapshot of the instant when the robot begins to decelerate to a velocity less than half the current velocity as it closes down on the vertex a marked in the figure. Evidently from the visibility polygon the vertex a casts a shadow and closer the robot gets to it slower needs to be the velocity.

Figure 13 is the time reduced counterpart of figure12. The snapshot is once again at a location close to the vertex a. Staying away from a permits nearly maximum velocity. The dip observed in the profile due to vertex a is negligible. Similarly staying from other vertices such as b allows for trajectory time of $21.79s$ when compared with $26.30s$ for figure 12. Modification of trajectory for better time-lengths proceeds along the lines of section 3.4. For the two

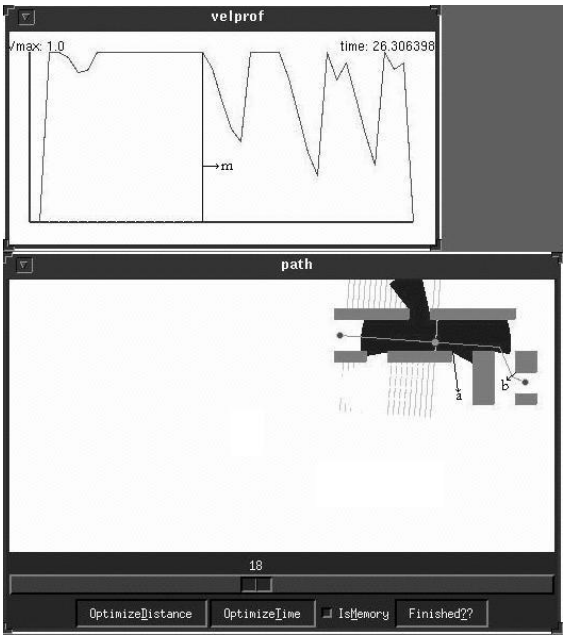

Figure 12: Path computed by a typical planner and its velocity profile shown on the top. The robot's velocity corresponding to its location on the trajectory is shown by a vertical line on the profile and labeled as m

examples discussed the robot's maximum acceleration and deceleration was fixed at $1m/s^2$, maximum velocity at $1m/s$ and the sensor range at $7m$. The maximum bound on the objects velocity was $1.5m/s$.

Figures 14 and 15 depict the planned trajectory and velocity profiles before and after reduction of trajectory time for our laboratory environment. The time reduced trajectory is shorter by more than 8 seconds as it widens its field of view by moving away from the bends while turning around them.

5.1 Effect of memorization on trajectory time

Figure 16 shows an environment with four corridors named $1, 2, 3$ and 4 with a path as computed by a non-holonmous planner.

Figure 17 shows the path obtained by minimizing time. It also portrays the robot's field of vision as it enters the corridor 3. The velocity profile for the above path is shown in figure 18. The location of the robot corresponding to its location on figure 17 is shown through the vertical line. The locations of the robot as it decelerates when its field of view of each of the corridor vanishes is also marked with the respective numbers on the profile.

Though the path of figure 17 is minimized in time its velocity profile still shows decelerations in the vicin-

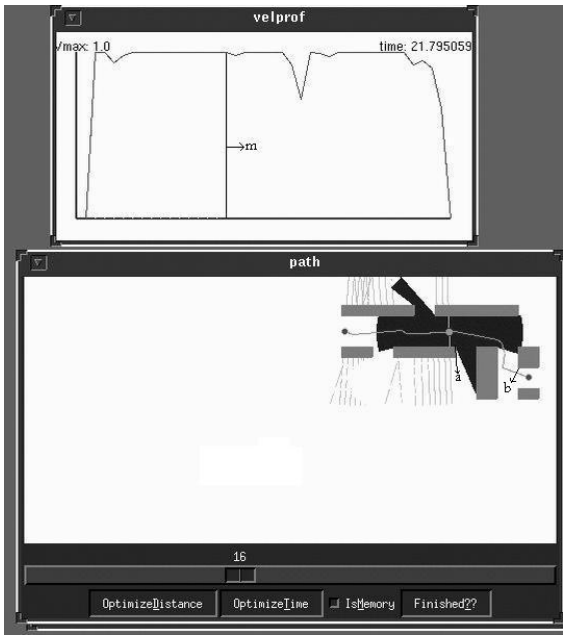

Figure 13: Path obtained after adaptation to reduced time-length

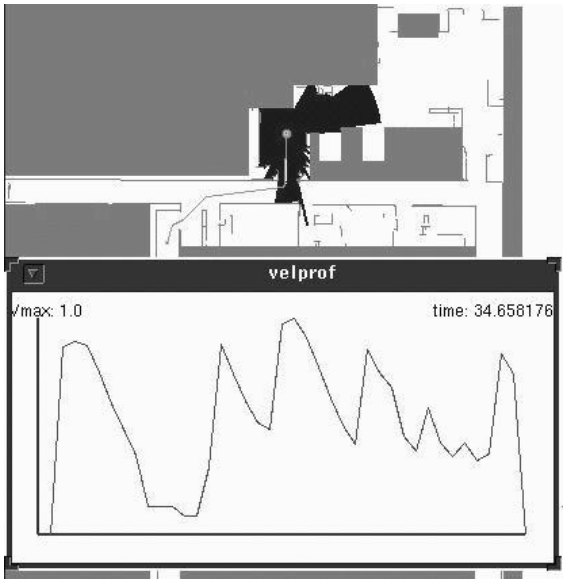

Figure 14: Planned trajectory before adaptation to a reduced time

ity of the corridors. This is due to the phenomenon discussed in section3.5 where the robot becomes blind to many parts of the environment it had seen at the preceding instant. Figure 19 shows the robot's field of vision at an instant after the instance shown in figure 17. There is a marked decrease in its field of vision at the latter instant that results in robot reducing its velocity in anticipation of moving objects from the blind zones depicted in the velocity profile.

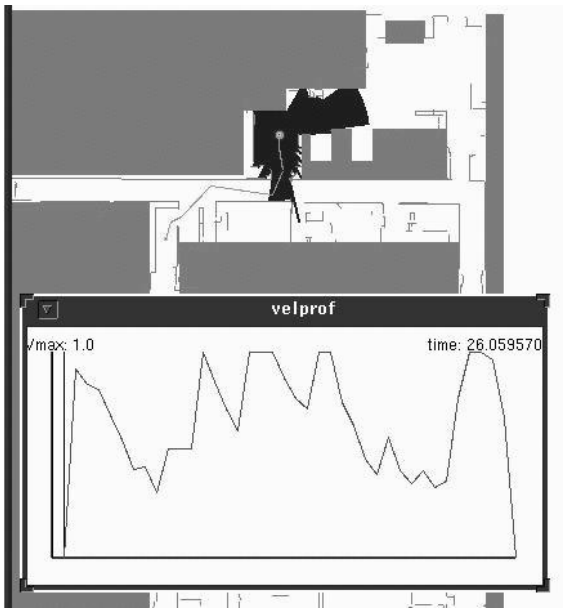

Figure 15: Time reduced trajectory at planning stage

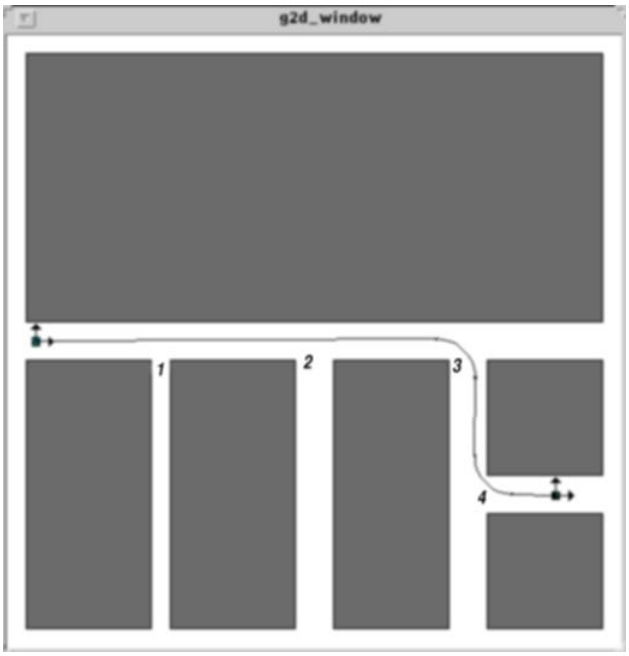

Figure 16: An environment with four corridors and a non-holonomous path

However when the robot is able to memorize its previous images the need to decelerate is nullified and the trajectory time further reduces. Figure 20 illustrates this where the decelerations shown in the velocity pro-

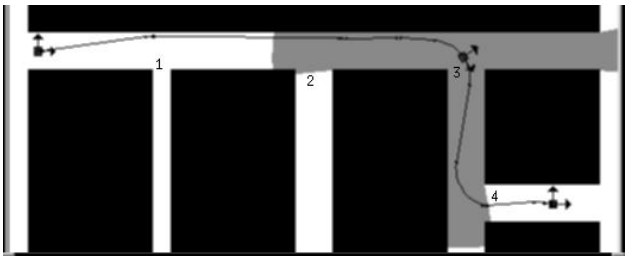

Figure 17: Robot's field of view as it enters corridor 3

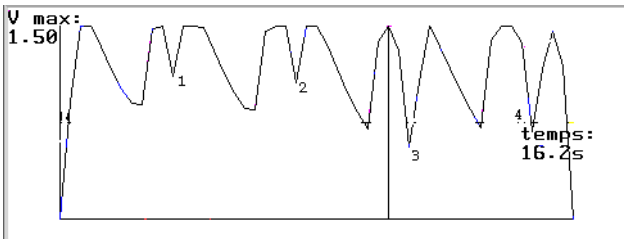

Figure 18: Velocity profile for the figure 17. Corresponding position of the robot shown in vertical line. Decelrations near the corridors are also marked with the same numbers

file of figure 18 at locations $1, 2, 3$ and 4 are now absent.

6 Simulation and Experimental Results at Execution Stage

6.1 Velocity profiles during plan and execution

In this section the velocity profiles obtained during planning and execution stage are compared in the absence of any new objects during execution. Figure 21 shows a simple planned trajectory and the corresponding velocity profile for our lab environment. Some of the obstacles are filled in gray and others are shown as segments (in gray). The robot is shown as a small circle and the star shaped polygon in black represents the field of vision of the robot at that location. The vertical line, marked m in the velocity profile represents the velocity of the robot corresponding to its position on the trajectory. The profile shows a subsequent drop in velocity, a consequent of robot getting closer to region marked, d, to which it is blind.

Fig 22 compares the planned and executed (in simulation) velocity profile. The executed trajectory tallied to a time of $12.28s$ in comparison with $12.25s$ for the planned profile. These figures illustrate that the executed profiles and execution times are close to the planned profiles and times while there are no changes

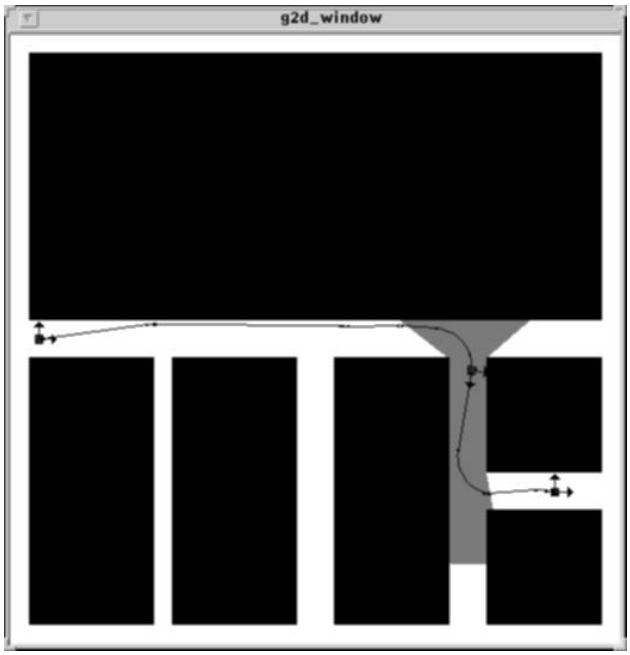

Figure 19: Robot's field of vision at an instant that immediately follows the instance of figure 17

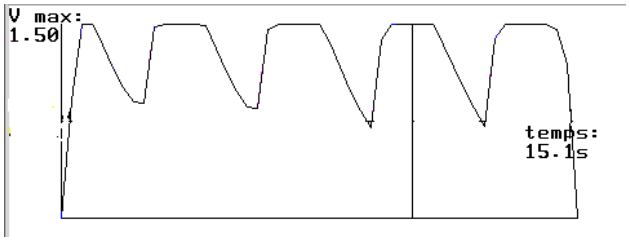

Figure 20: The velocity profile obtained after incorporation of memorization

in the environment.

Figures 24 and 25 show the execution by the Nomad $XR4000$ (fig 23) of paths computed by a standard planner. Figure 24 corresponds to the original path computed by the planner and figure 25 is its time reduced counterpart.

The velocity profiles during execution of the two paths are shown in figure 26. Some of the bigger drops in the unreduced profile are absent in the reduced profile as the robot avoids turning close to the obstacles that form the bends. The path of figure 25 got executed in $12.9s$ while the path in figure 24 was executed in $13.98s$. The figures are meant as illustrations of the theme that trajectories deformed to shorter timelengths at planning stage are also executed in shorter time during implementation than their unreduced versions.

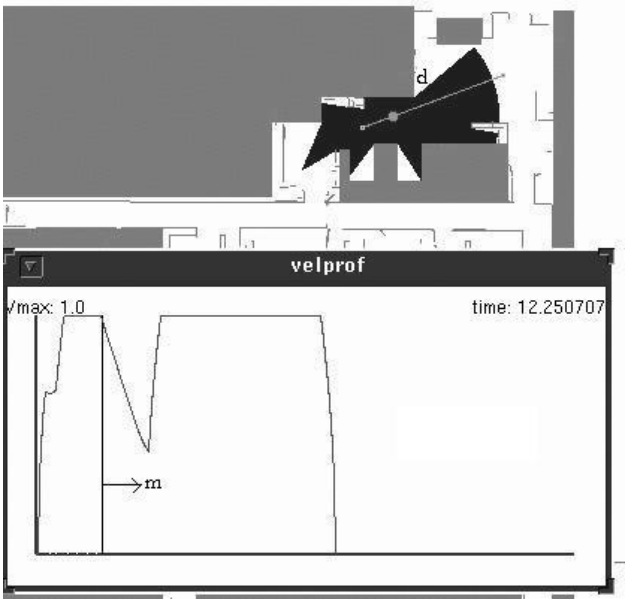

Figure 21: A simple planned trajectory and its velocity profile

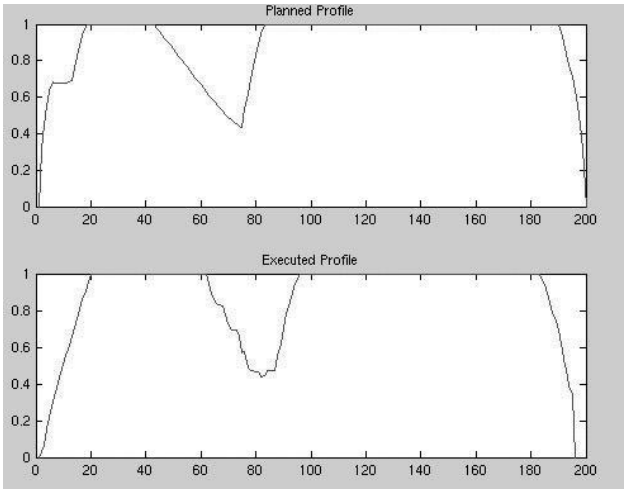

Figure 22: The planned and executed velocity profile in simulation

6.2 Online adaptation of paths for better trajectory time

This section presents results of the algorithm in the presence of newly added objects that affect the velocities of the robot in real-time. Figure 27 shows a path where the robot collision avoids the two new segments $S1$ and $S2$ intersecting the original planned trajectory but does not adapt its path for better time. The velocity profile for the same is shown in 28. Figure 29 is the counterpart of figure 30 where the robot adapts its

Figure 23: The Nomad XR4000 used in our experiments

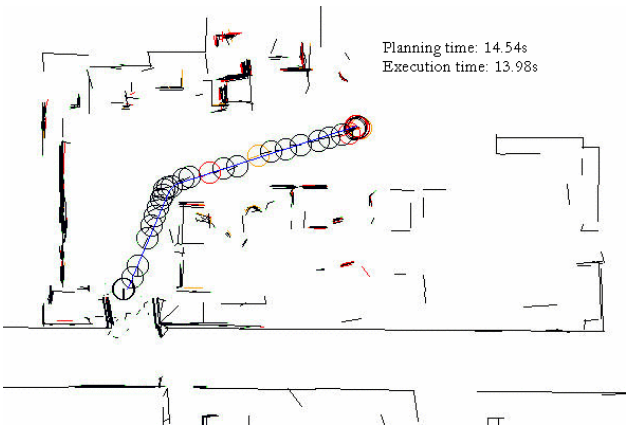

Figure 24: Execution of the original planned path by the Nomad

path to a better time-length reactively. The big dips in the velocity profile of figure 28 are filtered in figure 30 considerably as the robot avoids the obstacles with larger separation. The time reduced execution tallied to $10.9s$ while the unreduced version was executed in $12.5s$. The trajectory time at planning was $7.9s$. The above graphs are those obtained in simulation.

Figure 31 shows the unreduced executed path by the $XR4000$ Nomadic robot in our laboratory at LAAS. The obstacles in the original map are shown by black lines, while the segments perceived by the sick laser are shown in lighter shades of gray. Some of these segments get mapped to the ones in the map and the others are considered new segments. This is done by a segment based localization algorithm. The segments of concern here are those which form a box shaped obstacle marked B in figure 31. The vertex d of this

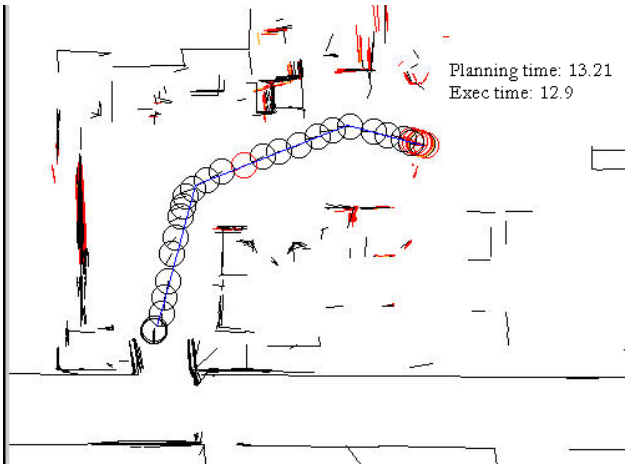

Figure 25: Execution of the time-reduced path by the Nomad

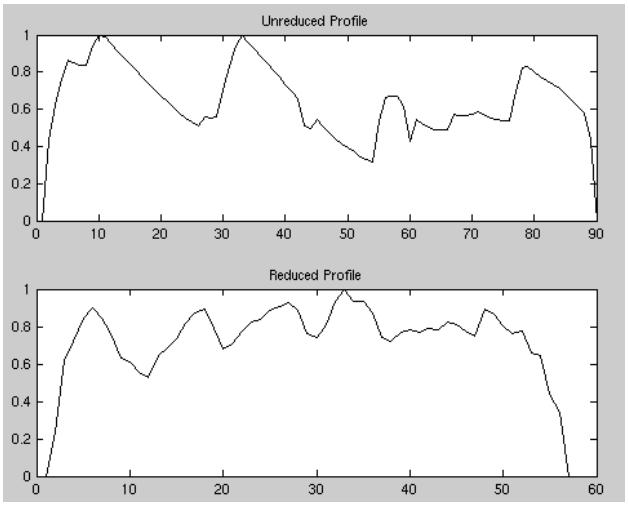

Figure 26: The top profile corresponds to the path executed in figure 24 and the bottom to figure 25

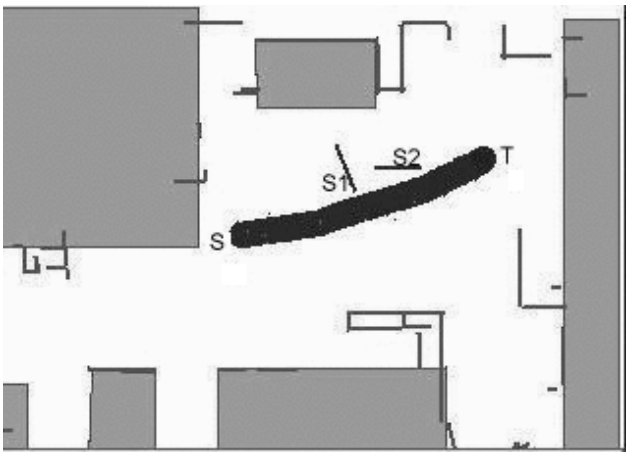

Figure 27: A simulated execution in the presence of two new segments $S1$ and $S2$ along with the corresponding velocity profile. The path is not adapted to better time-length. Start and goal locations marked as S and T.

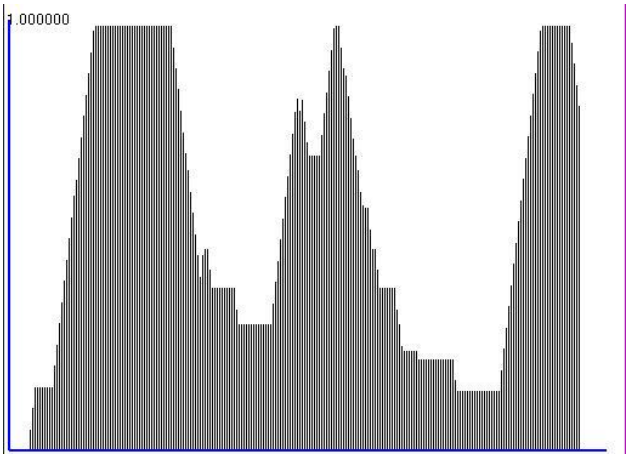

Figure 28: The velocity profile for the execution of figure 27

obstacle casts a shadow on robot's sensory field, which forces it to slow down at those locations due to equation 2. The execution time for this unreduced path is $10.6s$.

The time reuced counterpart is shown in figure32 that tallied to $9.6s$. The original planning time was $8.8s$ in the absence of the box shaped object.The velocity profile for the same is shown in figure 33.

7 Conclusions and Scope

A pro-active safe planning algorithm and its reactive version that facilitates real-time execution has been presented. The proactive nature of the algorithm stems from the computed velocity profile, $v_\tau(s)$, that guarantees immobility of the robot before collision with any of the possible mobiles that could interfere its future trajectory from regions blind to its sensor. The proactivity does not however come at the cost of robot's velocity or trajectory time. The knowledge of $v_\tau(s)$ computed over the trajectory $\tau(s)$ further facilitates reduction of the over all trajectory time $T(\tau)$ by adaptation of the initially planned path. Analysis of the scheme at the planning stage depict that the robot can have a velocity profile that achieves its maximum possible velocity for a sustained duration without many dips provided it stays away from doorways and narrow passages along its path. Memorization of previously cognized scenes also enhance the robot's performance through reduced trajectory time and a

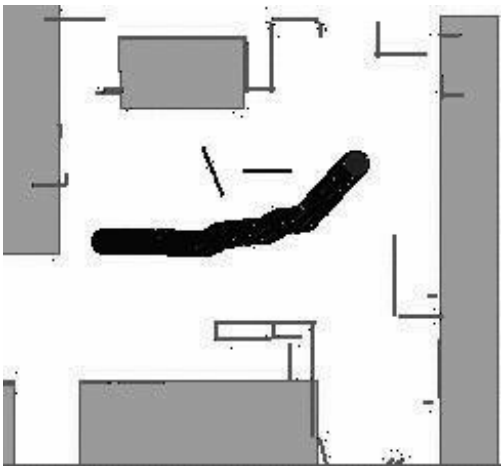

Figure 29: Path of figure 27 adapted to better time-length

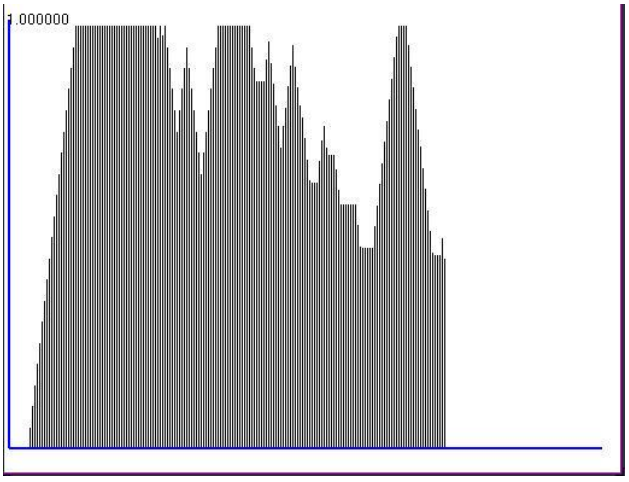

Figure 30: The velocity profile for the execution of figure 29

more uniform velocity profile.

A reactive extension of the scheme that facilitates real-time simulation and implementation is also presented. The scheme maintains the underlying philosophy of computing safe velocities and modification of paths for better trajectory time. Simulation and experimental results at real-time corroborate with our earlier results obtained at the planning stage (that by keeping away from vertices of objects that could hide mobiles the robot could move at higher velocities and obtain better time-lengths) and thus the efficacy of overall strategy is vindicated. The minimum distance over which the velocities need to be computed on the remaining trajectory during real-time such that the computed velocities are safe is theoretically established. This avoids repetetive computation of velocities over the entire remaining trajectory for every motion command, thereby reducing computational intensity and facilitating for real-time implementation. The methodology could be useful in the context of personal robots moving in areas where interference with mobile humans especially aged ones are generally expected.

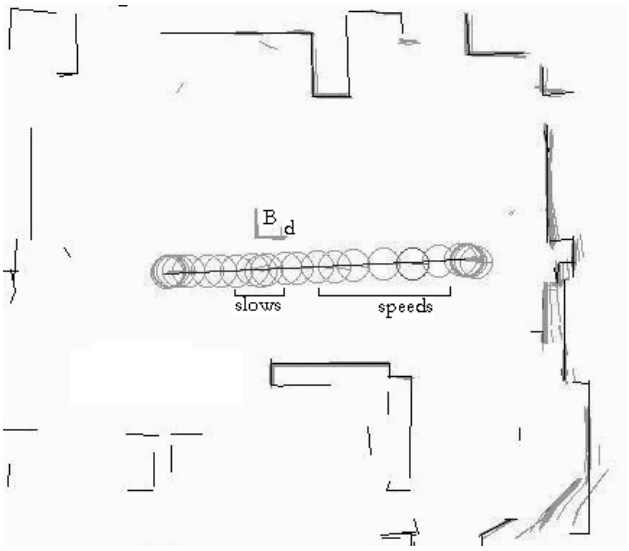

Figure 31: Unreduced path executed by the Nomad $XR4000$. The vertex d of the new box shaped object B forces a slow down near it.

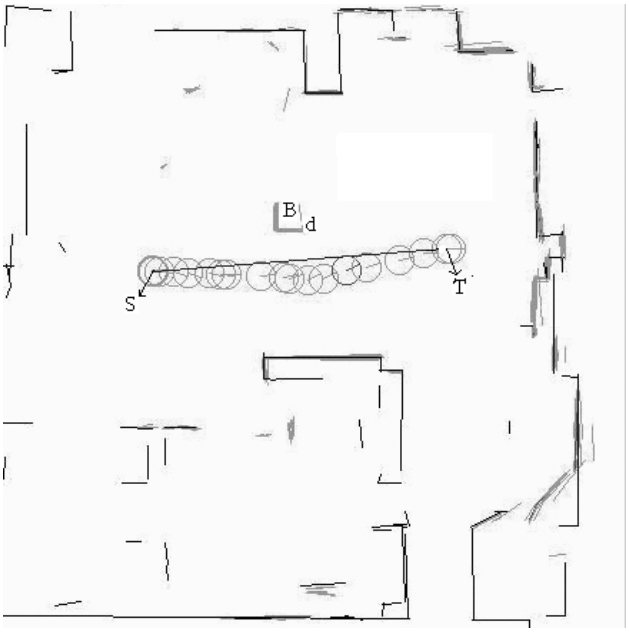

Figure 32: Time reduced path executed by the Nomad $XR4000$.Increasing linear and angular separation from vertex d facilitates a higher speed..

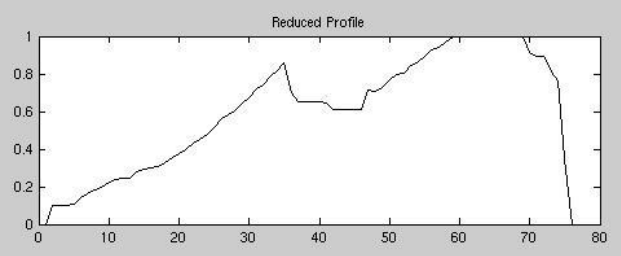

Figure 33: Velocity profile for the path executed by the Nomad in figure 32

Immediate scope of this work involves in incorporating the phenomena of memorization at the reactive level such that higher speeds are possible. The methodolgy needs to be validated in the presence of mobile objects that actually impinge on the path from blind zones with a provision for the robot to avoid the objects without halting continuing to respect safety considerations as well as minimizing trajectory time.

Acknowledgments

The work described in this paper was conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020 and by the French National Program ROBEA.

References

[1] R. Alami, T. Simeon, and K.Madhava Krishna. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Swizerland*, 2002.

[2] J.C. Alvarez, A. Skhel, and V. Lumelsky. Accounting for mobile robot dynamics in sensor-based motion planning: experimental results. *IEEE International Conference on Robotics and Automation, Leuven (Belgium)*, 1998.

[3] B. Bouily, T. Simeon, and R. Alami. A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. *IEEE International Conference on Robotics and Automation, Nagoya (Japan)*, 1995.

[4] D. Cruzel. Planification de mouvements sous contraintes de perception. Master's thesis, LAAS-CNRS, 1998.

[5] P. Fiorinin and Z. Schiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.

[6] S. Fleury, P. Soueres, and J.P. Laumond. Primitives for smoothing mobile robot trajectories. *IEEE Transactions on Robotics and Automation*, 11(3):441–448, 1995.

[7] M. Khatib, B. Bouily, T. Simeon, and R. Chatila. Indoor navigation with uncertainty using sensor-based motions. *IEEE International Conference on Robotics and Automation, Albuquerque (USA)*, 1997.

[8] K.Madhava Krishna, R. Alami, and T. Simeon. Moving safely but not slowly - reactively adapting paths for better trajectory times. *IEEE International Conference on Advanced Robotics, Quimbra, Portugal*, 2003.

[9] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[10] A. Lazanas and J.C. Latombe. Motion planning with uncertainty: a landmark approach. *Artificial Intelligence*, pages 287–315, 1995.

[11] J. Minguez and L. Montano. Nearness diagram navigation. a new real-time collision avoidance approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[12] N. Roy and S. Thrun. Motion planning through policy search. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Swizerland*, pages 2419–2425, 2002.

[13] Z. Schiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. *IEEE International Conf. on Rob. Automat.*, pages 3716–3721, 2001.

[14] C. Strachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Swizerland*, 2002.

[15] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. *ACM Symp. on Computational Geometry*, 1986.