Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

Predictive Control of Robot Velocity to Avoid Obstacles in Dynamic Environments

Amalia F. Foka and Panos E. Trahanias

Institute of Computer Science
Foundation for Research and Technology - Hellas (FORTH)
P.O. Box 1385, Heraklion, 711 10, Crete, Greece

and

Department of Computer Science
University of Crete
P.O. Box 1470, Heraklion, 714 09, Crete, Greece

{foka, trahania}@ics.forth.gr

***Abstract*— This paper introduces a methodology for avoiding obstacles by controlling the robot's velocity. Contemporary approaches to obstacle avoidance usually dictate a detour from the originally planned trajectory to its goal position. In our previous work, we presented a method for predicting the motion of obstacles, and how to make use of this prediction when planning the robot trajectory to its goal position. This is extended in the current paper by also using this prediction to decide if the robot should change its speed to avoid an obstacle more effectively. The robot can choose to move at three different speeds: *slow*, *normal* and *fast*. The robot movement is controlled by a Hierarchical Partially Observable Markov Decision Process (POMDP). The POMDP formulation is not altered to accommodate for the three different speeds, to avoid the increase of the size of the state space. Instead, a modified solution of POMDPs is used.**

I. INTRODUCTION

Navigation in dynamic, real-world environments is a complex and challenging task. Such environments are characterized by their complex structure and the movement of humans and objects in them. The robot has to avoid collision with obstacles and also reach its goal position in a fast and optimal manner. Obstacle avoidance is currently treated by methods that fall into two broad categories: *global* and *local* (or *reactive*). Global approaches (an overview of these methods can be found in [5]), have the advantage that they avoid obstacles with globally optimal paths but they cannot effectively deal with cases where unforseen changes in the obstacle movement occur. On the other hand, local obstacle avoidance methods [4], [1], [9], [3] can deal with unforseen changes in the environment but they treat the problem locally and in a suboptimal manner.

In our previous work [2], the obstacle avoidance problem was treated by making prediction of the obstacle's movement. Two kinds of prediction have been utilized: *short-term* and *long-term*. The short-term prediction refers to the one-step ahead prediction and the long-term to the prediction of the final destination point of the obstacle's movement. Both predictions are integrated into the POMDP model that is used to control the robot's movement. A hierarchical representation of POMPDs is used that enables us to solve the POMDP on-line at each time step. This approach, has the advantage that it is able to avoid obstacles in a globally optimal manner and also is able to deal effectively with unforeseen changes in the obstacle movement.

In this paper, our previous work is extended to allow the robot to decide if it should increase or decrease its speed to avoid an obstacle more effectively. Without control of its speed, the robot has to make detours or follow a suboptimal path to reach its goal in order to avoid collision with obstacles. In many cases, the robot can avoid making these suboptimal actions if it can either increase its speed to bypass the obstacle or decrease its speed to let the obstacle move away from the robot.

In the formulation of the problem with POMDPs we choose not to include the speed of the robot as a characteristic of its state, as with its location and orientation, or include speed actions in the action set. Such a choice would further increase the state space and make the memory requirements of the model impossible to manage. Instead, the solution of the POMDP is modified to account for the speed decision that the robot has to make.

The rest of this paper is organized as follows. In Section 2, the modelling of the problem is given. In Section 3, our methodology for controlling the robot's speed is explained. Experimental results are given in Section 4. Finally, in Section 5 a short discussion of the results of our approach concludes the paper.

II. PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES (POMDPS) AND PROBLEM FORMULATION

POMDPs are a model of an agent interacting synchronously with its environment. The agent takes as input the state of the environment and generates as output actions, which themselves affect the state of the environment.

A POMDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$ [6], that in our problem formulation is instantiated as:

- $\mathcal{S}$, is a finite set of states of the environment that are not observable. Each state in $\mathcal{S}$ corresponds to a discrete entry cell in the environment's occupancy grid map (OGM) and an orientation of the robot.
- $\mathcal{A}$, is a finite set of actions and consists of all possible rotation actions from 0^o to 360^o. The discretization of the action set depends on the actual levels of POMDP hierarchy (see later Section 2.1).
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ is the *state transition function*. $\mathcal{T}(s, a, s')$ is the probability of ending up in state s', given that the agent starts in state s and takes action a, $p(s'|s, a)$. $\mathcal{T}$ is defined according to the motion model of the robot.
- $\mathcal{Z}$, is a finite set of observations. The set of observations is instantiated as the output of the *iterative dual correspondence* (IDC) [7] algorithm for scan matching. At each time step, an observation is obtained by feeding the IDC with the current scan of the robot and a reference scan of the environment in which the robot operates. The IDC also requires an estimate of the robot's position from which the current scan was obtained, which is given as the robot's position before it performed the action. This position is taken to be the most likely state of the robot's belief state. It is reasonable to assume that the robot cannot move too far from its previous state at one time step. Therefore, the output of the IDC algorithm, that is the dx, dy and $d\theta$ from the estimated location provided, will be within certain limits. The output of the IDC algorithm is discretized and thus the set of observations remains manageable.
- $\mathcal{O} : \mathcal{A} \times \mathcal{S} \rightarrow \Pi(\mathcal{Z})$ is the *observation function*. $\mathcal{O}(s', a, z)$ is the probability of observing z, in state s' after taking action a, $p(z|s', a)$. $\mathcal{O}$, is again obtained according to the motion model of the robot.
- b_t, is the $belief$ the robot has at time t, that is a discrete probability distribution over the set of environment states, $\mathcal{S}$. Hence, $b_t(s)$ is the probability of the robot being in state s at time t, $p_t(s : s \in \mathcal{S})$.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the *reward function*, giving the expected immediate reward gained by the agent for taking each action in each state, $\mathcal{R}(s, a)$. $\mathcal{R}$, is built according to two occupancy grid maps (OGM): a *static* and a *dynamic*. The static OGM is built by calculating the distance of each cell to the goal position. The dynamic OGM is evaluated at each time step according to the short-term and long-term prediction. Long-term prediction refers to the prediction of the destination position of the obstacle's movement. Thus, the reward value of the grid cells that are in the trajectory from the obstacle's current position and its predicted destination position is discounted. The discount of a cell's reward value depends on the estimate of how far in the future this cell will be occupied.

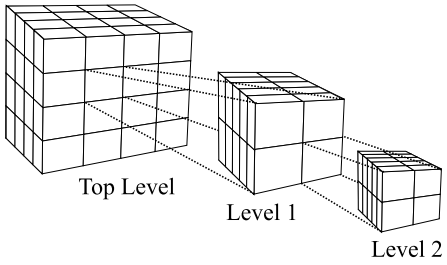


Fig. 1. The decomposition of states in the hierarchical POMDP.

A. Hierarchical POMDPs

Hierarchical POMDPs have been recently studied and two approaches have been proposed: state space hierarchy [10] and action space hierarchy [8]. Our approach to hierarchical POMDP formulation generalizes the above two approaches applying both state space and action space hierarchy. It decomposes a POMDP with large state and action space into multiple POMDPs with significantly smaller state and action space. The hierarchical POMDP has a tree structure, where going from the top level to the bottom, the resolution changes from coarser to finer and the number of POMDPs is increased. At the top level there is a single POMDP with coarse resolution and the basic four orientation angles (*North*, *South*, *East*, *West*). Subsequent levels are built by decomposing each state of every POMDP at a level to a new POMDP at the next level. At the bottom level, the states of every POMDP correspond to the states of the original flat POMDP. Fig. 1, illustrates the basic concept of hierarchical POMDPs. In this figure, it is shown how a state at the top level is decomposed to multiple states in subsequent levels.

B. Planning with Hierarchical POMDPs

Initially, the top level POMDP is solved. The solution of the top level POMDP gives intuitively the general route the robot should follow to reach the goal. Subsequently, the POMDP at the lower level that contains the robot's current state is solved with its goal being to follow the direction that the solution of the upper-level POMDP provided. In the same manner, POMDPs at all levels are solved until the bottom level is reached. The action obtained from the solution of the bottom level POMDP is the one that the robot will actually perform.

III. POMDP SOLUTION FOR CONTROLLING THE ROBOT'S SPEED

To solve a POMDP, the optimal policy that maximizes the infinite expected sum of discounted rewards from all states has to be evaluated. The optimal policy for each

state s at a time step t, $\pi_t^*(s)$, is evaluated as the sum of the immediate reward the robot will receive for the next action a it will perform and the expected discounted sum of rewards from all states until time step $t-1$. This is written as:

$$\pi_t^*(s) = \max_a \{\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')\pi_{t-1}^*(s')\},$$

where γ is a discount factor. The discount factor determines how important are the future rewards the robot will receive. If γ is zero, the robot will maximize the reward it will receive for the next time step only.

To be able to decide about the velocity, v, of the robot as well as the action, a, it should perform, the reward and transition functions should take into account the robot's velocity as well as the action. Hence, the above equation is rewritten as:

$$\begin{aligned} \pi_t^*(s) &= \max_{a,v} \{\mathcal{R}_m(s,a,v) \\ &+ \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_m(s,a,v,s')\pi_{t-1}^*(s')\}, \end{aligned}$$

where v is the velocity with which the robot will perform action a. $\mathcal{R}_m$ and $\mathcal{T}_m$ are the modified reward and transition functions, respectively, that take into account the robot's velocity as well as the action.

As mentioned above, the POMDP parameters are not altered to include information about the robot velocity. Therefore, we need a formulation of $\mathcal{R}_m$ and $\mathcal{T}_m$ that relates them to the original $\mathcal{R}$ and $\mathcal{T}$ functions that were built considering the $normal$ velocity of the robot only. This is achieved by introducing the notion of the *projected* state of the robot.

When the robot is in state s and performs an action a with a velocity v, other than the $normal$ velocity, it is expected to end up with high probability in a state s'. Then, the $projected$ state is defined as the state s_p, where if the robot executes action a, from state s_p, with its $normal$ velocity it will end up with a high probability to state s'. Of course, if v is the $normal$ velocity of the robot, then s_p is s. Fig. 2, illustrates how the projected state is determined.

Having defined the projected state, the relation of $\mathcal{R}_m$ and $\mathcal{T}_m$ to the original $\mathcal{R}$ and $\mathcal{T}$, respectively, can now be defined. By the definition of the projected state s_p, the relation of $\mathcal{T}_m$ to $\mathcal{T}$ is straightforward, and is written as:

$$\mathcal{T}_m(s,a,v,s') = \mathcal{T}(s_p,a,s').$$

The definition of $\mathcal{R}_m$ is not as straightforward as for $\mathcal{T}_m$. If $\mathcal{R}_m$ is defined as $\mathcal{R}_m(s,a,v) = \mathcal{R}(s_p,a)$, then the robot would always choose to move with the $fast$ speed.

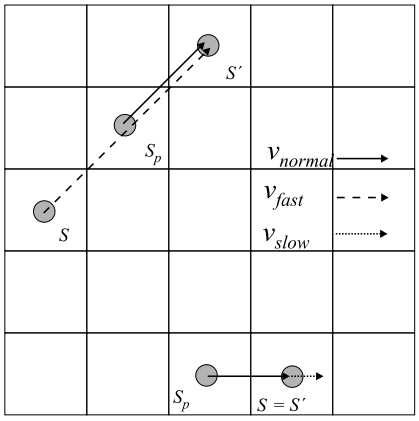


Fig. 2. Definition of the projected state s_p.

This is because the $fast$ speed will always get the robot closer to the goal and thus the reward that it will receive will be bigger. Instead, it is desirable that the robot moves at a different speed from its $normal$ speed only if it has to avoid an obstacle. For that reason change of speed is penalized.

The penalty factor for change of speed is defined in relation to the reward function to ensure that the robot has the desirable behavior. The reward function is built by calculating the distance of each grid cell to the goal position. The value assigned to each grid cell is the distance to the goal position, inverted and normalized. Therefore, the reward value of neighboring cells will always differ by a certain amount, as it can be seen in Fig. 3(a). When the static OGM is built, the average expected difference of the reward value between adjacent grid cells for every action a, $Ediff(a)$, can be evaluated. For example, in Fig. 3(a), it can be seen that for action $North$ the difference of the reward values is always 0.125. Hence, $Ediff(a)$ will be used as a penalty value.

Thus, we define $\mathcal{R}_m$ as:

$$\mathcal{R}_m(s,a,v) = \mathcal{R}(s_p,a) - \alpha \cdot \frac{|v - v_n|}{v_n} \cdot Ediff(a),$$

where v_n is the $normal$ velocity of the robot and α is a constant that controls how preferable are the velocity changes. The bigger the value of α, the less preferable the velocity changes are. The $\frac{|v-v_n|}{v_n}$ factor, ensures that when v is the $normal$ velocity, the reward the robot will receive will not be penalized. It also accommodates for the effect of the difference between the $fast$ or $slow$ velocity with the $normal$ velocity on the reward the robot receives by $\mathcal{R}(s_p,a)$. For example, when the $fast$ velocity is double the $normal$ velocity, then we expect that the reward the robot will receive in these two cases will differ by $Ediff(a)$. In the case that the $fast$ velocity is triple the $normal$ velocity, then $\frac{|v-v_n|}{v_n}$ will be equal to 2, as

we expect that the reward the robot will receive in these two cases will differ by $2 \times Ediff(a)$.

When there is no obstacle in the route of the robot to its goal, the reward values will be unaltered and dependent only on the distance to the goal. Hence, in the case of the $fast$ velocity, the reward the robot will receive after being penalized, will be the same with the reward the robot will receive for choosing the $normal$ velocity, for α equal to 1. In the case that there is an obstacle moving in the route of the robot to its goal, then the reward values of the cells that are predicted to be occupied by the obstacle in the future will be discounted. Then, the reward the robot will receive for $fast$ velocity will be bigger than the reward for $normal$ velocity even after it is penalized for changing speed.

An example for this case is illustrated in Fig. 3(b). It is assumed that there is an obstacle moving in the environment. The reward value of the cell that corresponds to the obstacle's current position is set to zero. The reward value of the cells that are in the trajectory from the obstacle's current position to its predicted destination position (the shaded cells in the figure) is discounted. The original reward value of these cells can be seen in Fig. 3(a). The discount of the reward values is not the same for all cells, since it depends on the estimate of how far in the future each cell will be occupied. The robot is currently at the cell denoted with s. If the robot moves with the $normal$ velocity, the maximum reward it will receive is 0.375 for action $North$ or $East$. When the reward values for the $fast$ velocity are evaluated, it can be seen that the reward for executing action *North-East* will be 0.5. That is because the robot will end up in the state denoted with s', and the reward it will receive is the reward for executing action *North-East* from the state denoted with s_p with the $normal$ velocity, minus the expected difference in the reward values for action *North-East*. Hence, the robot will move with the $fast$ velocity and bypass the moving obstacle.

In the case of the $slow$ speed, the reward the robot will receive for executing any action from the projected state will always be smaller than the reward the robot will receive for choosing the $normal$ speed, when there is no obstacle. This reward will be further decreased by the penalty factor. Therefore, for the robot to choose the $slow$ speed, the reward it receives for the $normal$ and $fast$ speed has to be smaller. This will be the case when there is an obstacle very close to the robot and did not have a long-term prediction to be able to avoid it by increasing its speed.

0.5	0.625	0.75	0.875	1
0.375	0.5	0.625	0.75	0.875
0.25	0.375	0.5	0.625	0.75
0.125	0.25	0.375	0.5	0.625
0	0.125	0.25	0.375	0.5

(a)

0.5	0.625	0.75	0.875	1
0.375	0.4	0.625	0.75 s'	0.875
0.25	0.375	0.3 s_p	0.625	0.75
0.125	0.25 s	0.375	0.2	0.625
0	0.125	0.25	0.375	0

(b)

Fig. 3. (a) The static OGM and (b) An example of the robot choosing to move with the $fast$ velocity.

A. Belief Update

The belief the robot has for being in a state s, is updated every time the robot executes an action a, and obtains an observation z. The belief the robot has at time t that it is in a state s, $b'(s)$, after taking action a and making observation z, is the probability $P(s|z,a,b)$, where b is the belief the robot has for all states at time $t-1$. The belief update can be expressed in terms of the observation and transition functions of the POMDP, according to Bayes rule as shown in the equation below.

$$\begin{aligned} b'(s) &= P(s|z,a,b) \\ &= \frac{P(z|s,a,b)P(s|a,b)}{P(z|a,b)} \\ &= \frac{P(z|s,a,b)\sum_{s'\in S} P(s|a,b,s')P(s'|a,b)}{P(z|a,b)} \\ &= \frac{P(z|s,a)\sum_{s'\in S} P(s|s',a)P(s'|b)}{P(z|a,b)} \\ &= \frac{\mathcal{O}(s,a,z)\sum_{s'\in S} \mathcal{T}(s,a,s')b(s')}{P(z|a,b)} \end{aligned}$$

As before, this equation has to be expressed in terms of a velocity v, as well as an action a. Thus, it is rewritten as:

$$b'(s) = \frac{\mathcal{O}_m(s,a,v,z)\sum_{s'\in S} \mathcal{T}_m(s,a,v,s')b(s')}{P(z|a,b)}.$$

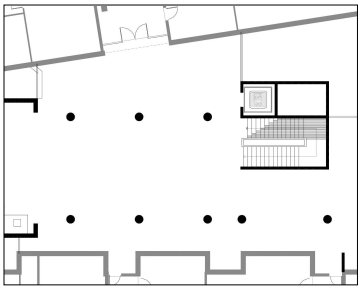


Fig. 4. The main entrance hall of the FORTH building where the experiments were conducted.

The relation of $\mathcal{O}_m$ to the original observation function $\mathcal{O}$, using the projected state s_p, is straightforward and is written as:

$$\mathcal{O}_m(s, a, v, z) = \mathcal{O}(s_p, a, z)$$

.

Finally, the belief update equation becomes

$$b'(s) = \frac{\mathcal{O}(s_p, a, z) \sum_{s' \in S} \mathcal{T}(s_p, a, s') b(s')}{P(z|a, b)}.$$

IV. RESULTS

The experiments were performed in the main entrance hall of the FORTH building, an area of approximately $250m^2$, shown in Fig. 4. The robot had to reach the goal position that was given and avoid people walking in the environment. Many experiments have been conducted and all confirmed the performance of the proposed method. Here we present sample results (see video) that illustrate its effectiveness.

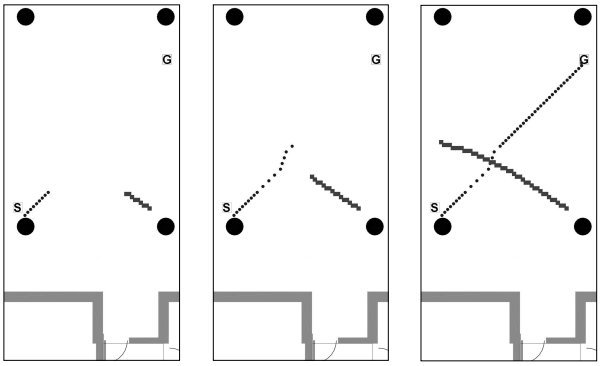
The environment's area was discretized in cells of $10cm^2$. The action angles and orientation of the robot were discretized in steps of 10^o. The total number of states of the robot was approximately 1 million. To be able to solve the POMDP in real time the hierarchical representation of POMDPs, explained in Section 2 of this paper, has been used with 4 levels.

The robot's $normal$, $fast$ and $slow$ velocity was set to $1m/sec$, $2m/sec$ and $0.5m/sec$, respectively.

In the example figures, points marked with "S" and "G" are the start and end position of the robot respectively. The robot's position at each time step is shown with a circle point and the obstacle's position is shown with a rectangle point.

A. Avoiding obstacles with $fast$ velocity

In the example case, shown in Fig. 5, the robot has to reach the goal position but there is an obstacle moving that blocks its trajectory. The robot starts moving in the optimal direction to reach the goal position but at the time step shown in Fig. 5(a), the robot has the prediction of



(a) $t = 9$ (b) $t = 17$ (c) $t = 46$

Fig. 5. The robot moves at $fast$ speed to avoid an obstacle.

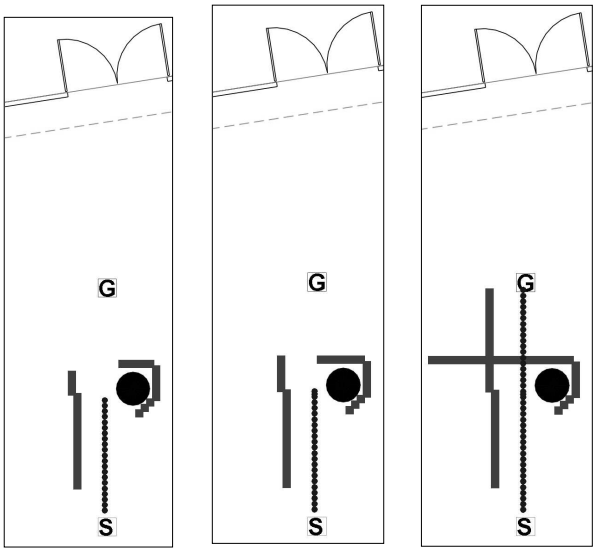
the obstacle's movement that its going to block its route. As shown in Fig. 5(b), the robot starts moving with the $fast$ velocity to bypass the obstacle. After the robot has passed the area that was predicted to be occupied by the obstacle, it reverts to the $normal$ velocity to continue its movement to the goal position. With this behavior, the robot eliminated the need of making suboptimal actions to make a detour in order to avoid the obstacle.

B. Avoiding obstacles with $slow$ velocity

In the example, shown in Fig. 6, the robot has to simply move in the $North$ direction to reach the goal position. There is an obstacle moving parallel to the robot and another one moving around the column of the building. The robot starts moving in the $North$ direction as expected to reach the goal position. At the time step shown in Fig. 6(a), the robot detects the obstacle that was moving around the column and makes the prediction that it is going to block its direction. The obstacle is close to the robot and hence it cannot increase its speed to bypass it. In addition, the robot cannot move to the left or the right because on the one side there is the second obstacle while on the other side there is the column. For that reason, its slows down until the obstacle moves away from its path, as shown in Fig. 6(b). Following, it reverts to its $normal$ speed to continue its movement towards the goal position. Enabling the robot to decide to slow down, it prevented making the decision to make a detour by turning to the right where it would come too close to both obstacles.

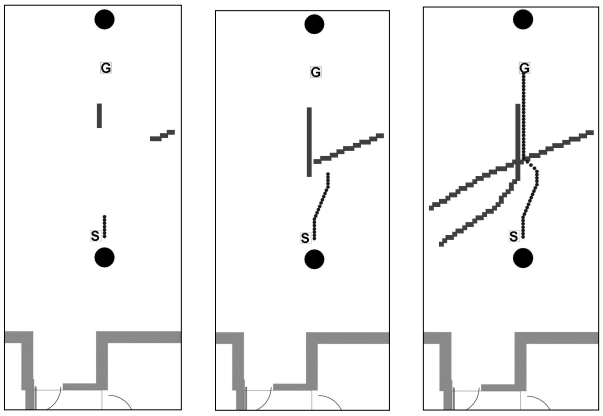
C. Avoiding obstacles with a detour

In the last example, we demonstrate a case where the robot cannot make a safe decision to move with either the $fast$ or the $slow$ velocity to avoid an obstacle. In such cases, the robot is able to decide to make a detour. In the example case, shown in Fig. 7, there is an obstacle



(a) $t = 21$ (b) $t = 23$ (c) $t = 41$

Fig. 6. The robot moves at $slow$ speed to avoid an obstacle.



(a) $t = 7$ (b) $t = 21$ (c) $t = 51$

Fig. 7. The robot makes a detour to avoid an obstacle.

that blocks completely the optimal trajectory to the goal and another one that moves towards the optimal trajectory. The robot makes the prediction of the movement of both obstacles and estimates that it cannot move with the $fast$ or $slow$ velocity, since one of the obstacles is moving towards itself, as shown in Fig. 7(a). Hence, it decides to make a detour to avoid both obstacles.

V. CONCLUSIONS AND FUTURE WORK

In this paper it is proposed to use prediction techniques within a probabilistic framework to control the robot's velocity to avoid obstacles in dynamic environments. The decision about the robot's speed is made by modifying the solution of the POMDP instead of adding the speed actions into the action set of the model. This choice prevented the further increase of the memory requirements of the model. Future work involves considering having as speed actions $increase\ speed$ and $decrease\ speed$ instead of having three standard speeds. The prediction of the obstacle's velocity will also be considered to be able to control the robot's velocity more effectively.

VI. ACKNOWLEDGMENTS

This work was partially supported by EC Contract No IST-2000-29456 (WebFAIR http://www.ics.forth.gr/webfair) under the IST Programme.

VII. REFERENCES

[1] J. Borenstein, and Y. Koren. "The vector field histogram - fast obstacle avoidance for mobile robots", *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, 1991, pp.278–288.

[2] A. Foka and P. Trahanias. "Predictive autonomous robot navigation", in *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[3] D. Fox, W. Burgard and S. Thrun. "The dynamic window approach to collision avoidance", *IEEE Robotics & Automation Magazine*, vol. 4, no.1, 1997.

[4] O. Khatib. "Real-time obstacle avoidance for robot manipulator and mobile robots", *International Journal of Robotics Research*, vol. 5, no. 1, 1986, pp.90–98.

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

[6] M. Littman. *Algorithms for Sequential Decision Making*, PhD thesis, Department of Computer Science, Brown University, 1996.

[7] F. Lu and E. Milios. "Robot pose estimation in unknown environments by matching 2D range scans", *Journal of Intelligent and Robotic Systems*, vol. 18, 1998, pp.249–275.

[8] M. Montemerlo, J. Pineau, N. Roy, S. Thrun and V. Varma. "Experiences with a mobile robotic guide for the elderly", in *Proceedings of the International Conference on Artificial Intelligence (AAAI 2002)*, 2002.

[9] R. Simmons. "The curvature-velocity method for local obstacle avoidance", in *Proceedings of 1996 IEEE International Conference on Robotics and Automation (ICRA)*, 1996.

[10] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. "Learning hierarchical partially observable markov decision processes for robot navigation", in *Proceedings of 2001 IEEE International Conference on Robotics and Automation (ICRA)*, 2001.