Proceedings of the 14th International Conference on
Control Systems and Computer Science
Bucharest, Romania, July 2003

SEARCHING A TARGET WITH A MOBILE ROBOT

Adriana Tapus **Olivier Aycard**

Autonomous Systems Lab
Swiss Federal Institute of
Technology Lausanne (EPFL)
CH – 1015 Lausanne

INRIA Rhones-Alpes –GRAVIR
ZIRST – 655 avenue de l'Europe
38330 Montbonnot.
France

Adriana.Tapus@epfl.ch ; Olivier.Aycard@inrialpes.fr

Abstract: This paper presents an application of the Markov Decision Processes to search a target with a mobile robot. The robot does not know the absolute position of the target. It only knows the position of the target relative to its position. Markov Decision Processes have been widely used in mobile robotics. In this paper, we show experimentally that they are well suited for our task. Moreover, we explain how the mobile robot plans and re-plans a path to the target taking into account the incomplete knowledge it has on the target. Some simulation experiments are given.

Keywords: Planning, Markov Decision Processes, Markov Localization, Mobile Robotics, Simulation.

1. INTRODUCTION

Mobile robots often need to operate in domains that are only incompletely known or change dynamically. In this case, they need to be able to re-plan quickly as their knowledge changes. For instance, mobile robots might observe obstacles in the terrain that they did not know about or their goal locations might change (for example, if they chase moving targets). Since re-planning from scratch is often extremely time consuming, a solution is planning methods that are incremental versions of heuristic search methods. For instance, Koenig and Likhachev (2001) propose a heuristic search method that repeatedly determines a shortest path from the current robot coordinates to the goal coordinates while the robot moves along the path and discovers its environment. But, in this approach uncertainties on actions and observations are not taken into account. Markov Decision Processes have been used to model uncertainty for mobile robot planning and navigation, see (Cassandra, *et al.,* 1996). Moreover, they permit incremental planning, but they have been relatively little used in domains that are only incompletely known or change dynamically.

In this paper, we will present a new application of the Markov Decision Processes to a mobile robot task where the environment is incompletely known: searching a target in an indoor office environment. In this task, the mobile robot does not know the absolute position of the target. It only knows the position of the target relative to its position. Moreover, at the beginning of the task, the mobile robot does not know its own position. During the task it will always have an incomplete knowledge of its own position (as it is always the case in robotics). It will always have only an incomplete knowledge of the absolute position of the target. During the task, it will plan and re-plan its path to the target taking into account its dynamic and incomplete knowledge on the absolute position of the target.

This paper is organized as follows: after briefly describing the Markov Decision Process in the next section, we will explain our model in the third section. Section 4 is the description of our methodology. We

present some experiments and discuss results in section 5, and give some conclusions and perspectives in section 6.

2. THE RESEARCH MODEL

In this section, the stochastic model known as Partially Observable Markov Decision Processes (POMDPs) will be briefly presented. This model was developed in the operations research community and has been introduced to artificial intelligence researchers; see (Cassandra, *et al.,* 1994).

2.1 Markov Decision Processe

A Markov Decision Process (MDP) is defined as a tuple <S,A,T,R>, where S is a finite set of environment states which is identified by the robot; A is a finite set of actions; T is the transition function between the environment states based on the action performed; and R is the *reward function,* which permits us to determine the objectives and the possible dangerous zones. The notation T:SxAxS→[0,1] is used, where **T(s,a,s')** is the probability to go from state s to state s', when action a is performed and **R(s)** for the reward of the robot in state s.

Using an MDP, the robot knows at each instant its position (current state). The actions must provide all the information for predicting the next state.

A policy π is a mapping from S to A, specifying an action to be taken in each situation. The optimal policy is calculated as a function of the reward function R. The value of state s∈S, $V_\pi(s)$ (given a policy π), is the expected value of the sums of the reward function to be received at each future time step. In the literature, the two most important criteria for calculating the optimal policy are: the *average-reward criterion,* which it is interested only in an immediate reward, and the *discounted infinite horizon reward*. The last criterion is the most widely used and it is very interesting because it permits a compromise between the immediate reward and the reward in the distant future. The expected reward is formulated as follows:

$$V_\pi(s) = E\left(\sum_{t=0}^{\infty} \gamma^t \cdot R_t\right) \qquad (1)$$

where R_t is the reward received at the t-*th* step of executing policy π after starting in state s. The discounted factor, $0 \le \gamma \le 1$, controls the influence of the reward in the distant future. If $\gamma = 0$, the value of a state is determined only by rewards calculated on the next step. We generally set the value of γ to be close to 1, because we are interested in problems with a larger horizon. The definition of V can be rewritten as:

$$V_\pi(s) = R(s) + \gamma \, \Sigma_{s' \in S} \, T(s, \pi(s), s') \times V_\pi(s') \qquad (2)$$

We say that a policy π dominates (is better than) π' if, for all s∈S, $V_\pi(s) \ge V_{\pi'}(s)$, and for at least one s∈S, $V_\pi(s) > V_{\pi'}(s)$. A policy is optimal if it is not dominated by any other policy. Given a Markov Decision Process and a value of γ, it is possible to compute the optimal policy fairly efficiently, as described in Puterman (1994).

The two most important algorithms used to calculate the optimal policy are: Value Iteration, developed by Bellman (1957) and Policy Iteration, firstly described by Howard (1960). By calculating the optimal policy we are planning the paths from all the points to the destination.

2.2 Adding Partial Observability

As we saw in the previous section, in the MDPs the state is not completely observable and a model of observations must be added. Thus, a finite set O of possible observations and an observation function O will be added. We write OS (o,or,s) for the probability of making observation *o* being in the state *s* and having orientation *or*. A belief state is a discrete probability distribution over the set of environments states, S, representing for each state the robot's confidence that it is currently occupying that state. We write Pr_{t-1} (s) the probability value that the robot is in environment state s at the time t-1. To solve the problem of the state being not completely observable, we generally use a set of belief states, which contains the set of the most likely states (MLS), see (Cassandra, *et al*., 1996), in which the robot can be. This set of belief states can be determined from the previous belief state Pr_{t-1}, the previous action *a,* and the data coming from the robot's sensors (the observations of the robot), as follows:

$$Pr_t(s', or) = \frac{OS(o, or, s') \times \sum_{s \in S} Pr_{t-1}(s) \times T(s, a, s')}{\sum_{s \in S} Pr_t(s)} \qquad (3)$$

where $Pr_t(s)$ is a normalizing factor. The resulting function ensures that the current belief state accurately summarizes all available knowledge.

3. DESCRIPTION OF THE MODEL

The model described in this paper uses an abstract Bayesian model of the environment and the robot's actions and observations. In this section, the robot's actions, observations and the information received about the target will be described.

3.1 Environment

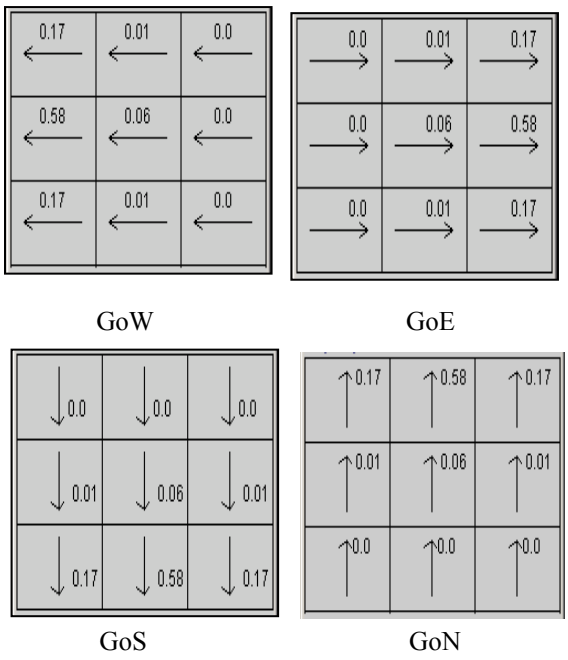
The robot is in an a priori or previously learned map, and its position is unknown. The only information available is that the robot is somewhere on the map. If no a priori map is available there are many applications that build such a map when the robot explores its environment.

3.2 Abstract Actions

The robot has four abstract actions: goN, goS, goW and goE. The action model specifies, for each state and action, the probability that each state will occur. These probabilities were obtained through informal experimentation, although it would be a fairly simple extension to make the robot learn from its own experience.

Table 1 shows the action probabilities used in our experiments, i.e. the transition function.

Table 1: **Action probabilities for abstract actions**



The transition function is very important in the MDPs because due to it the uncertainty of the actions is taken into account. The nine cases in the pictures above represent the position of the robot (the central case) and its eight adjacent positions. The arrows show the action that can be done in the transition. For example, in the case of goW action, 0.06 (the central case) means that, with the probability 0.06, the robot would stay in the same case. Also in the case of goW action, 0.58 means that, with the probability 0.58, the robot will go west independently of its orientation.

3.3 Abstract Observations

In each state, the robot is able to make an abstract observation. The observation model specifies, for each state and action, the probability that a particular observation will be made. Thus, in this model the orientation of the robot is taken into account.

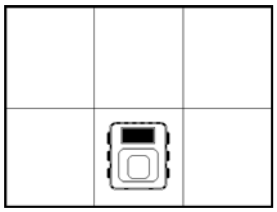


Figure 1: **Physique zone of observations used**

Five zones of observations, as depicted in figure 1, are used. These five zones correspond to the adjacent states of the position of the robot. The robot will tell us if what it sees is an obstacle or a free case, with the help of its sensors. A score, i ($0 \le i \le 5$) is calculated and for each possible score a probability, **Pr(i|5)**, is associated. The score is the number of zones in which the observation of the robot *o* is the same with the real observation in the state *s*, when the robot is oriented to *or*. This score is calculated for each zone. The observation function that has been implemented is:
Pr (5 / 5) = 0.5 ; Pr (4 / 5) = 0.05 ; Pr (3 / 5) = 0.02
Pr (2 / 5) = 0.004 ; Pr (1 / 5) = 0.0019 ; Pr (0 / 5) = 0.0005

3.4 Information about the target

The task is to reach a target of an unknown position. At the beginning the initial position of the robot is unknown and the only information available is the robot's observations and some data, which is transmitted by the robot's sensors. The data tells us how far the target is with respect to the position of the robot.
An axis with the origins in the coordinates of the robot, axis, directed downwards (see figure 2) was chosen.

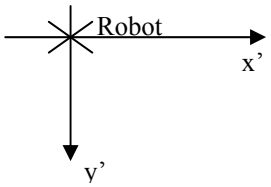
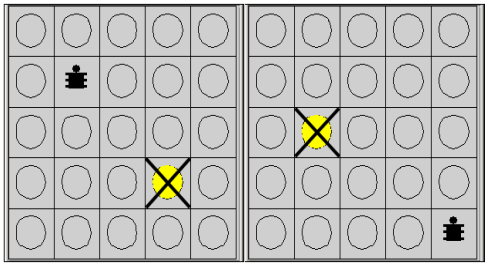


Figure 2: Axis for the localization of the target relatively to the robot's position

In the pictures below (see figure 3) the gray locations represent free spaces and the cross blobs represent the objectives.



a *b*

Figure 3: Positioning of the target compared to the robot position

The figure 3a shows that the target is 2 cases on x'-axis and 2 cases on y'-axis away from the robot's position. Negative values can be used too, i.e. as the figure 3b shows it, the target is to -3 cases on x'-axis and -2 cases on y'-axis compared to the robot position. In other words, the Manhattan's distance between the target and the robot's position is calculated. To calculate the Manhattan's distance two transceivers can be used. One is on the robot and the other one is on the target. The information transmitted is the distance between them and the direction of the target.

4. DESCRIPTION OF THE METHOD

The method is divided in four parts: the observations, the planning, the execution of a plan and the elimination of some hypothesis about the target. The robot will make an observation and will do an action, and after that a distribution over the states is calculated. Information about the Manhattan's distance between the target and the robot's position is also available. At each step, an iteration of the planning and an iteration of the execution of a policy will be done. Each time, a policy is computed, it is sent to the execution process that drops the previous one and starts working on the current one. This permits the robot to act before the optimal policy is calculated.

Such a technique can be developed, because the planning and execution use iterative algorithms. The utilization of an iterative algorithm in the planning allows re-planning only by reusing and improving the previous policy. At each step, a distribution over the possible positions of the robot in the environment is calculated. This distribution and the information detained about the target will be used to make some hypothesis about the possible locations of the target. A reward function, which takes into account hypothesis, will be used: hypothesis with strong (resp. weak) probability will have high (resp. weak) reward. At the last part of the method, hypotheses about the target with a very weak reward will be eliminated.
In the following paragraphs, the approach will be examined and explained with the help of a detailed example. Figure 4 shows the type of environments that were chosen.

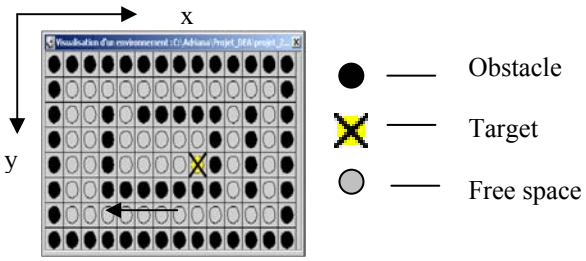


Figure 4: Synthetic office environment

5. SIMULATION EXPERIMENTS

The parameters used will be indicated before starting to explain into more detail the approach. For planning, the performance is measured as the discounted sum of rewards (γ =0.9999) starting from the initial state. The robot starts off with no information about its starting state. When there is more than one start state for a given experiment, the robot starts randomly in one of the states, with its set of belief states initialised to the uniform distribution over the set of possible start states. To better understand the example, we give on a purely informative basis the position and the real direction of the robot. Thus, the robot is in the box corresponding to case 3 on x-axis and case 6 on y-axis, oriented towards west (see the arrow on figure 4).

The Observations

At the first step, the initial position of the robot is unknown and thus we will have an uniform distribution. Each time the robot will make new observations a new distribution will be calculated. The distribution and the information on the target will help to calculate a new distribution of the possible locations of the target.

Planning and re-planning

At the beginning, the position of the target is unknown. The only information about the target is the data coming from the robot's sensors. Therefore, a target will be put in each possible destination position. The destinations are the states in which the robot can probably be. Thus a "propagation" of the targets on the map will be done.
A reward function described as follows: the obstacles will have a reward of (-cst) / (1-γ) and the free states a reward equal to –cst, is used. We multiplied by a factor

cst to make the actual targets very attractive (strong probability) and the others very little attractive (weak probability). Thus the targets are absorbent states. An arbitrary value for cst, in our case equal to 5 has been chosen.

Consider (x_s, y_s) the data coming from the robot's sensor. If the robot is in a state (x,y) with a probability p, the reward that we give to the state (x_s+x, y_s+y) so that this one is a target is of (p-1)*cst. Thus several targets with different rewards will be distributed on the map. The distribution of the possible locations of the target will be given by the reward function R. If R ≈ 0, there is a great probability that there is the target and if R ≈ -cst the probability that we found the destination is very weak.

In this manner hypothesis on the target can be modeled, by giving a larger reward to the most probable hypothesis on the target. An iteration of the planning algorithm Policy Iteration, with knowledge of this distribution, is made. Now all this can be described on an example. At the first iteration, after having collected the observations, and the data from the sensors, the corresponding targets were propagated on the map (see figure 5). Once the possible hypotheses are available on the map, a step of planning is executed. This step of planning gives us the various ways that might lead us to the target.

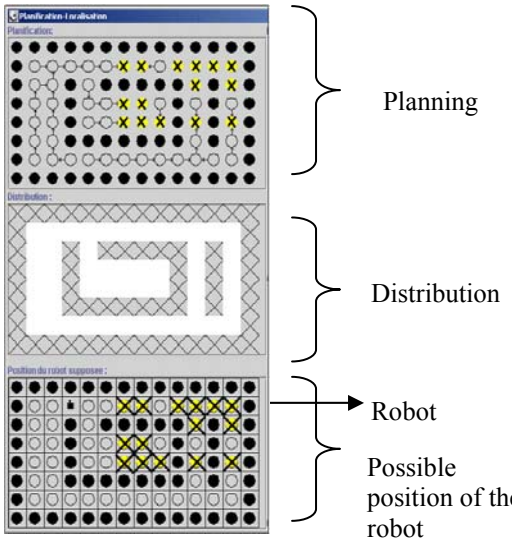


Figure 5: The first iteration of the algorithm

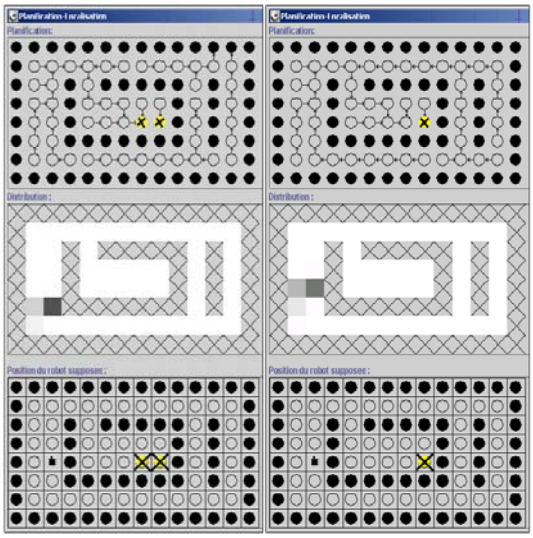
During the first iteration, the hypotheses that correspond to improbable positions of the robot are eliminated, and a re-planning step is done after this. The targets states removed become free states. The robot takes the observations and the data from the sensors, and it executes a new policy by taking into account all the changed parameters. Even if at the beginning we didn't know almost anything about the position of the robot and about the real destination, as we advance in time the incertitude is reduced. Due to the data we get from the sensors and the observations of the robot, the big number of hypotheses from the beginning decreases.

Execution

After the planning step is finished, a certain action is done. For our tests, we choose to do the action associated to the most likely state (MLS). Each time the robots moves new information about the target's position relatively to the robot's position is available.

Elimination of some hypotheses

During the phase of planning, the hypotheses that have a weak reward (very close to -cst, weak reward) will be eliminated, and the hypothesis with the strong reward (nearest to 0, strong reward) will be kept. When a target is removed, it is replaced by a free state and thus it will have a reward of –cst (see figure 6). When only one hypothesis (corresponding to the real position of the target) is remaining on the map, the planning and the execution will be executed like we have described before.



Step 4 Step 5

Figure 6: Ameliorations

Results

We tested this approach in simulation on different type of environments. We performed 25 missions on synthetic office environments**.** A mission is classified as successful when the robot reaches the real target point on the map. We had a success rate of 92%. The two experiments where the robot doesn't arrive to the real target are due to the noise that we have introduced in the observation and in the Manhattan's distance

between the robot and the target. The algorithm we have presented proved to be quite good.

6. CONCLUSIONS AND OUTLOOK

This paper presents an application of the Markov Decision Processes to search a target with a mobile robot. The mobile robot plans and re-plans a path to the target taking into account the incomplete knowledge it has on the position of the target. We show in simulation that our method is well suited for this task. Our method can be extended, without any modification, to unknown environment and to moving target. Moreover, it can be combined with the chase of an elusive target by a mobile robot equipped with a vision system, as described in (Coue and Bessiere, 2001). In this case, our method will be used to search the target and the mobile robot will chase the target when it is visible by the vision system.

Future work will focus on the complete testing of all the components presented in this paper together in a single implementation.

REFERENCES

Bellman, **R** (1957). Dynamic Programming. *Princeton University Press.*

Cassandra, A.R., **L.P. Kaelbling** and **M. L. Littman** (1994). Acting optimally in partially observable stochastic domains. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA.

Cassandra, A.R., **L.P. Kaelbling** and **J. A. Kurien** (1996). Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In: *Proceedings of IEEE International Conference on Intelligent Robots and Systems*.

Coue, C. and **P. Bessière** (2001). Chasing an Elusive Target With A Mobile Robot. In: *Proceedings of IEEE International Conference on Intelligent Robots and Systems.*

Dean, T., L. Kaelbling, J. Kirman and **A. Nicholson** (1993). Planning with deadlines in stochastic domains. In: *Proceedings of the 11th National Conference of Artificial Intelligence*.

Howard, R.A (1960). Dynamic Programming and Markov Processes. *MIT Press*, Cambridge, Massachusetts.

Koenig, S. and **M. Likhachev (**2001). Improved Fast Replanning for Robot Navigation in Unknown Terrain. *Technical Report*. GIT-COGSCI-2002/3, College of Computing, Georgia Institute of Technology, Atlanta (Georgia).

Puterman, M.L (1994). Markov Decision Processes. *John Wiley & Sons*, New York.