Intelligent Rescue robots:

Implementation of MDP and Collision avoidance for Multi Robots

Niyousha Rahimi
Graduate student, Mechanical Engineering dept.
University of Washington
Seattle, WA, USA
nrahimi@uw.edu

Abstract—Markov Decision Processes is among the most popular algorithms in robotics for indoor robot navigation problems. They allow computing optimal policies in order to achieve a given goal, accounting for actuators' uncertainties. In this work, MDP has been implemented for two homogenous robots in an indoor environment, the goal of which is to take two victims out of a corrupted building, autonomously, without collision to obstacles or to one another. Three different policies are derived for this purpose. Robot's trajectories for each policy, plots of value functions and the results of simulation are presented.

Keywords; : Markov Decision Process, Multi Robots, Navigation, Path Planning

I. INTRODUCTION

To rescue people from a corrupted indoor environment using autonomous robots, the robots have to localize themselves in the environment and navigate through obstacles. Path planning has been the subject of several studies [1, 3, 4], some of which consider uncertainty in the environment, in the dynamics of the Robots and sensors. This process allows the robot to get from one point to another in the shortest path, without collision to any obstacles in its way. Path planning algorithms are measured by their computational complexity, the feasibility of real-time motion planning and the accuracy of implementing uncertainties [4]. Compared to the traditional robot navigation approaches, Markov Decision processes (MDP and POMDP) allow to explicitly represent the various forms of uncertainties in navigation problems: actuator and sensor uncertainties, dynamic environments, uncertainty about the initial position of the robot, etc. A MDP policy focuses on the uncertainty of the dynamic of the robot and its actuators, generating plans with high success probability [2].

The most exquisite research on MDP is presented in [1]. This research represents two state aggregation techniques for reducing the number of states for driving policy, in order to increase tractability in large environments. These methods decrease the amount of time needed for driving optimal plans. Moreover, [2] uses Reactive Navigator to solve the problem of dealing with unknown obstacles while driving a policy.

However these two researches only consider the uncertainty in the dynamics of the robots and actuators, not the noise of the sensors, assuming that the sensors are perfect. In contrast, [5] explicitly model actuators and sensors uncertainty by implementing Partially Observable Markov Decision Process (POMDP). While the robot is uncertain about its position, it maintains a probability distribution over its current pose and it would never completely be lost. This approach is more realistic in the real world. The main difference of this work to the others is that instead of driving policies, it only uses stochastic processes to localize the robot.

In the present work MDP is implemented for two homogenous robots in an indoor environment to drive optimal paths through obstacles. The policies we obtained lead the robots through efficient paths to the goals, no matter where the start point is in the environment. The main originality of our approach is to implement MDP on two multi robots while considering collision avoidance of these two with each other. One of the robots would be considered as unknown dynamic obstacle for the other one and while they reach a defined distance toward one another, one would stop until the other one passes.

In the following section the main idea of MDP and its formulas are presented. Technical approach and implementation is discussed in section 3 and results are given in section 4. Finally, the last section contains concluding remarks.

II. MARKOV DECISSION PROCESS [6]

In a perfect world, there is no uncertainty. The robot simply knows its initial pose, the location of the goal and its exact position at each time. Moreover, the actions have predictable effects and can be pre-planned. However, the real world is full of uncertainty. The robot's sensors have noises which results in perception errors, and the actuators may cause actions different than what is planned. An alternative approach to solve these problems is MDP. This method deals with stochastic action effects by assuming that the action model $p(x' \mid u, x)$ is non-deterministic, although the environment can be fully sensed and the perceptual model $p(z \mid x)$ is bijective. This method assumes that the effect of a given control action may be uncertain but once the action is executed, the robot has perfect sensing to determine its exact position. Hence, MDP take into account the uncertainty of the dynamics of the robots and

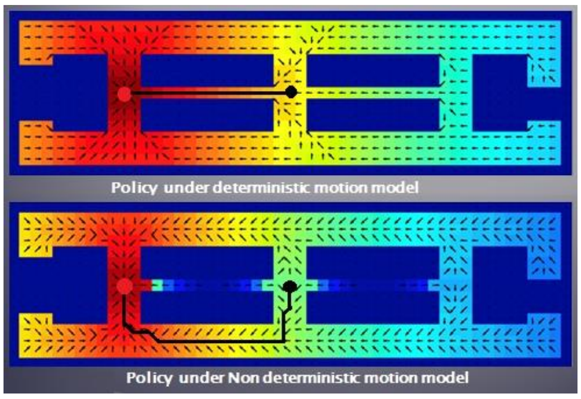

Fig. 1. The value function and control policy for a MDP with (a) deterministic and (b) non-deterministic action effects

actuators and ignore sensor errors. By driving a policy, this method allocates a control action to each state of the environment that the robot might encounter. In other words, a policy is mapping from state space to control space. Planning in a nun-deterministic environment helps the robot avoid paths that could cause collision with obstacles. As it is shown in *Fig.1* driving policy under non-deterministic motion model leads the robot through a long but safe path. In this figure control actions at each state are shown with an arrow, leading the robot to the goal.

A Markov Decision Process models an agent which interacts with its environment, taking as input the states of the environment and the location of the goal and generating actions for each state as outputs. MDP framework can be defined as follows:

- $\mathcal{X}$ is a finite set of states of the environment;
- $\mathcal{U}$ is a finite set of actions;
- $\mathcal{P}: \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow [0, 1]$ is the state transition probability $p(x' \mid u, x)$ which indicates the probability of moving to state "x' " when action "u" is executed in state "x".
- $r: \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ is a payoff function that gives the cost or reward of executing control action "u" at state "x".

MDP algorithm needs a defined state environment to drive policy. After that, an optimal policy π defines for each state of $\mathcal{X}$, an optimal action to be executed. In this paper we used **Value Iteration** algorithm to drive optimal policy. Value iteration recursively calculates the utility of each action relative to a payoff function. Every policy has an associated value function which measures the expected value of this specific policy. This value function is obtained by *Eq.1*.

$$\tilde{V}(x_i) = \gamma \max_u [\, r(x_i,u) + \sum \tilde{V}(x_j)\, p(x_j \mid u, x_i)] \qquad (1)$$

An optimal policy is the one that maximizes the value function at each step. Hence, the policy would be defined by Eq.2.

$$\pi(x_i) = \operatorname{argmax}_u \; [\, r(x_i,u) + \sum \tilde{V}(x_j)\, p(x_j \mid u, x_i)] \qquad (2)$$

γ is a discounting factor used to give more or less importance to future rewards.

The overall algorithm of MDP using Value iteration with finite state (x_i in $\mathcal{X}$) is as follows:

```
1. For i=1 to N do
      $\tilde{V}(x_i) = r_{min}$
   endfor
2. Repeat until convergence
      For i=1 to N do
         $\tilde{V}(x_i) = \gamma \max_u [\, r(x_i,u) + \sum \tilde{V}(x_j)\, p(x_j \mid u, x_i)]$
         $\pi(x_i) = \operatorname{argmax}_u [\, r(x_i,u) + \sum \tilde{V}(x_j)\, p(x_j \mid u, x_i)]$
      end for
   end repeat
3. Return $\tilde{V}$
4. Return $\pi$
```

Based on the algorithm above, firs minimum of value function is assigned to each state and then recursively value function is calculated until convergence.

III. Technical Approach and Implimentation

A. *Problem Statement*

The objective of this work is to design two automated homogenous robots that rescue two people in a corrupted environment. By having the map of the environment, i.e. the exact location of all obstacles and exit doors, and the location of two victims and assuming that the robots have perfect perception of the environment, we want our robots to get to the victims (each victim is a goal for each robot) and get them out of the building through exit door (exit location is the second goal for both robots). Given this objective, we would drive three policies for each goal location, using MDP. It should be noted that for the purpose of avoiding collision of two robots with each other, if the distance between two robots reaches a certain value, one of them would stop until the other one passes.

B. *Model Used*

In the current work we used two circular shaped robots that are 0.8 meter in radius (*Fig.2*), and can execute four control

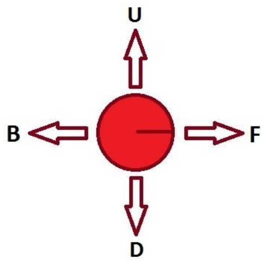

Fig. 2. Circular shaped robot with four control actions

actions:

- Going Forward
- Going Backward
- Going Up
- Going down

These robots operate in a 2D environment that is 52×82.9 square meter.

For modeling the states we used Histogram Filter, which would divide the environment into 52×82.9 grid points. Moreover we used Binary Occupancy Grid to determine obstacles. If a grid point is occupied by an obstacle the map would return 1, otherwise it would be 0.

C. *MDP Properties*

For implementing MDP we need to define State Probability and Payoff Function.

1) *State Probability*

We assumed that the actuators of the robot have uncertainty and the probability that the robot correctly peruse a given control action is 80%. There is 20% chance that the robot would end up in the neighboring states (10% each). Therefore, state probability for a given control action would have non zero value for only three states.

2) *Payoff Function*

In MDP algorithm for path planning, robotic action selection is defined by reaching a goal configuration while simultaneously optimizing the cost. In this research we defined pay of function for a given control action in three scenarios (*Fig.3*):

1. If the next state gained by pursuing this control action is a state occupied by an obstacle, the payoff function would penalize the robot by -1000

   $r(x,u) = -1000$

2. If the next state gained by pursuing this control action is closer to the goal, than the previous state, the payoff function would reward the robot by

   $r(x_i,u) =$ by $+1000/(\text{norm}(x_{new} - x_{Goal})+2)$

   We defined this value so that as the robot gets closer to the goal configuration, the payoff function rises.

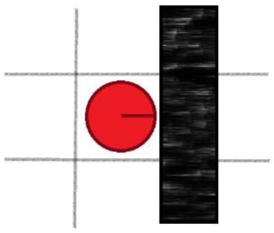

Fig. 3. The robot in the environment encountering an obstacle

3. Otherwise

   $r(x_i,u) = 3$

3) *Discounting factor and Value iteration Convergence*

As the equation (1) and (2) showed, we need a discounting factor to evaluate future rewards. Our experiments showed that $\gamma = 0.9$ would be a suitable choice. Moreover, in MDP algorithm value function would be calculated for all the states in the environment several times until its values converge. This happens when the span of the value function (given by the difference between the max and min component values) at the current and the previous iteration is less than a user-defined threshold, ε. Experiments show that better policies are gained with very small ε. In this study we chose ε to be 0.01.

D. *Implementation and Simulation*

In the present work, MDP algorithm for path planning is implemented in a 2D simulation environment in MATLAB that is 52×82.9 square meter. The simulation is executed on two robots in a known, fully observable environment. As it is stated before, we used two circular shaped robots and in order to take into account the size of the robot and avoid any possible collision with obstacles, we inflated the map and treat the robots as point objects (*Fig.4*).

The location of the goal and the starting point for the first and second robot and the location of exit are given below:

$X_{Goal1} = [60;18]$, $X_{Goal2} = [40;40]$, $X_{Exit} = [88;12]$

$X_{start1} = [5;38]$, $X_{start2} = [1;34]$

The overall algorithm of our method is as follows:

1. Calculating Value Function and Policy for the first Robot (Goal1).
2. Calculating Value Function and Policy for the second Robot (Goal2).
3. Calculating Value Function and Policy for Exit point.
4. Simulation of the two robots reaching the goals simultaneously without collision to obstacles or one another.
5. Simulation of the two robots reaching the exit point.

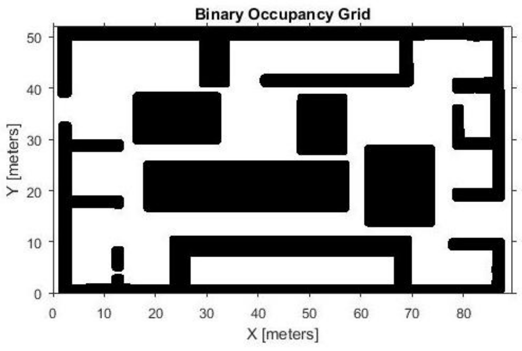

Fig. 4. Inflated Map

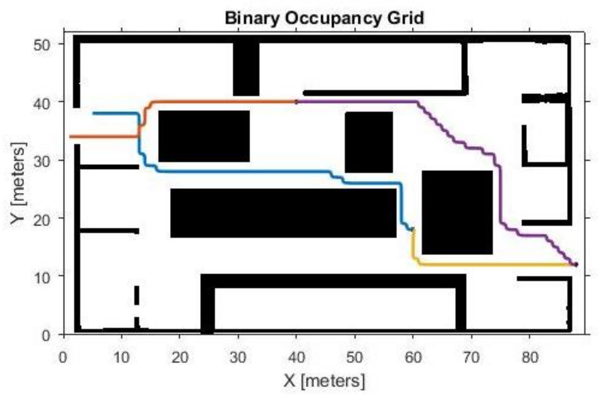

Fig. 5. Simulation of the robots at final point

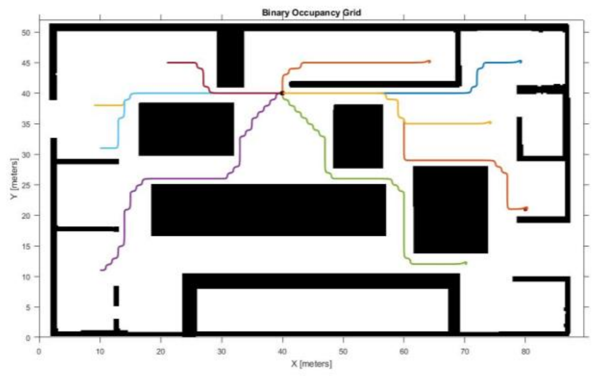

Fig. 7. Paths to the goal from different initial conditions for the second goal

IV. Results

Fig.5 shows the simulation of the robots at the final position. Starting simultaneously at initial positions, when the robots reach a distance of 5m from each other, sensors will detect the presence of a dynamic obstacle hence, the second robot stops, while the firs one passes.

Moreover, our simulation for each policy with different initial positions proved that the MDP algorithm drives optimal control actions for each state of the environment, resulting in convergence of paths to the goal, for initial positions that are in close neighborhood. *Fig.6* and *Fig.7* presents several paths with different initial conditions for our first and second robots. It should be noted that by setting Ɛ = 0.01, we gained highly efficient policies. As it is shown in *Fig.6* the path starting from [81;22] or path with initial position [86;31] correctly surpass the obstacles on their way and reach the goal, while if we chose a higher value for Ɛ, the path derived by the policy will not lead to the goal. Instead the robots got stuck behind the obstacles.

Furthermore, *Fig.8* evidently demonstrates the effect of our chosen payoff function. The states that are closer to the goal coordinate have higher value functions and the states that are occupied with obstacles have the lowest value.

This figure presents the core idea of Markov Decision Process based on value iteration; in which by rewarding or penalizing the robot when executing a given control action, optimal paths to a given goal is achieved. Value function at each state of the environment shows the position of the obstacles and the states leading to the goal. The alternative policies for different goals in an environment can be validated by comparing value functions for three goals (*Fig.8*, *Fig.9* and *Fig.10*).

V. Conclusion

In this study, Markov Decision process has been implemented for path planning of two robots in a fully observable environment. A suitable payoff function is established for MDP algorithm and collision avoidance of the robots is tested. Moreover, collision avoidance of the two robots with each other has been taken into account considering the robots to be dynamical objects.

Additionally, by comparing different trajectories of a policy and different value functions of each policy, the optimality of this method is confirmed. Our experiment showed that the value of "Ɛ" has a delicate influence on policy's efficiency.

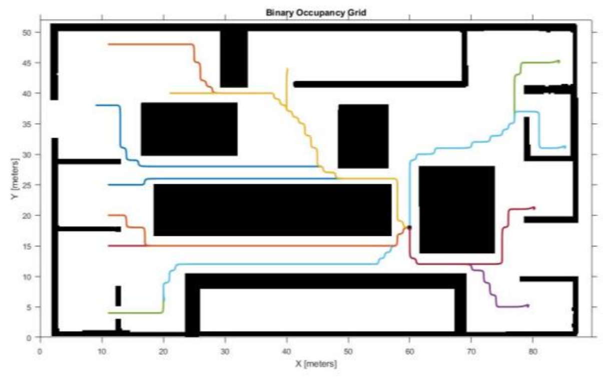

Fig. 6. Paths to the goal from different initial conditions for the first goal

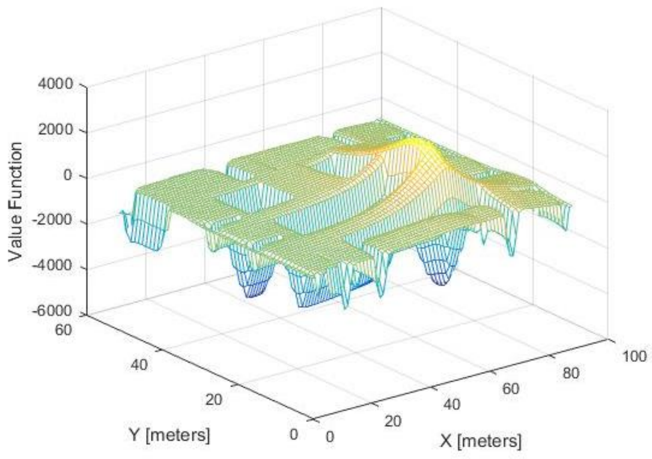

Fig. 8. Value Function for the first goal

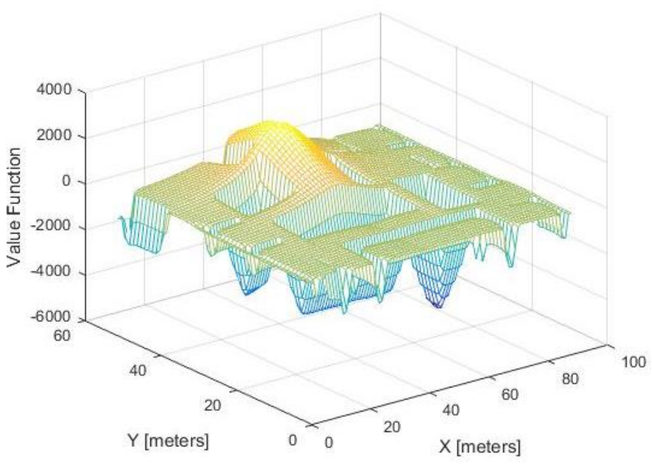

Fig. 9. Value Function for the second goal

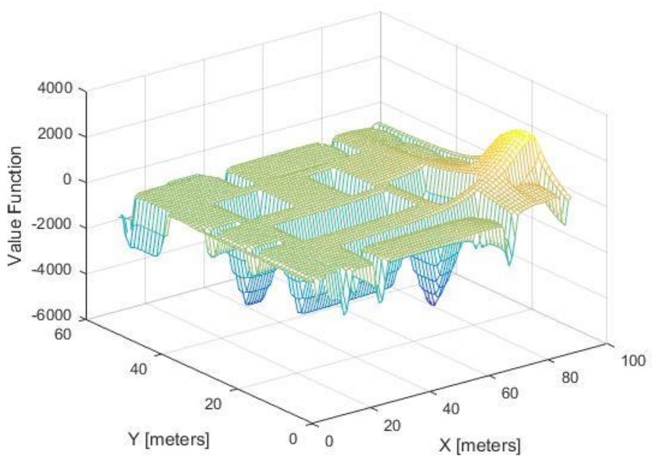

Fig. 10. Value Function for the third goal

References

[1] Pierre Laroche and François Charpillet. "State Aggregation for Solving Markov Decission Problems, An Appicaton of Mobile Robotics" LORIA INRIA-Lorraine, U.M.R. 7503 Campus scientifique, B.P. 239

[2] Pierre Laroche Franc¸ois Charpillet Ren´e Schott, "Mobile Robotics Planning using Abstract Markov Decision Processes" LORIA INRIA-Lorraine, U.M.R. 7503 Campus scientifique, B.P. 239 - 54506 Vandoeuvre-l`es-Nancy, FRANCE

[3] Stergios I. Roumeliotis, Ioannis M. Rekleitis, "Analysis of Multirobot Localization Uncertainty Propagation" , Department of Computer Science & Engineering, University of Minnesota, Mechanical Engineering, Carnegie Mellon University.

[4] S. M. Masudur Rahman Al-Arif, A. H. M. Iftekharul Ferdous, Sohrab Hassan Nijami, "Comparative Study of Different Path Planning Algorithms: A Water based Rescue System" Department of Electrical and Electronic Engineering, Islamic University of Technology (IUT) Boardbazar, Gazipur – 1704, Bangladesh

[5] Sven Koenig and Reid G. Simmons, "A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models", Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213-3890.

[6] Sebastian Thrun, Wolfram Bugard, Dieter Fox, "Probabilistic Robotics" The MIT Press, Cambridge, Massachusetts, London, England.