

Hierarchical Gaussian Reinforcement Learning for Path Planning in Uncertain Environments

Nouar AIDahoul, Zaw Zaw Htike, Rini Akmeliawati, Amir Akramin Shafie, Sheroz Khan

*Department of Mechatronics Engineering, Faculty of Engineering,
international Islamic University Malaysia*

Abstract

Most of the issues in planning and controlling of robots are caused by uncertainties in the actuators and sensors of robots. Path planning is of paramount importance for autonomous mobile robots. This paper presents a path planning approach that is based on hierarchical Gaussian reinforcement learning. This approach differs from traditional Q-learning in two ways: its ability to deal with continuous states and actions and its ability to work in uncertain (nondeterministic) environments. We propose a path planning algorithm for robots in uncertain environments by using hierarchical Gaussian Q-learning. We used Matlab to perform experiments in simulation. The simulation experimental results seem suggest the efficiency of the proposed algorithm in finding optimal paths of autonomous agents.

Keywords: reinforcement learning, path planning, hierarchical Q-learning, Gaussian function, autonomous underwater vehicle.

1. Introduction

Path planning under uncertainty is very important for improving the robustness of robotic systems [1]. There has been significant progress recently on robot motion planning algorithms that deal with robot control uncertainty, sensing uncertainty, environment changes and Model uncertainty.

In general, the environment is stochastic. That leads to performing same action in the same state may lead to different rewards or different next states [2]. Above that, the state transitions probabilities or getting specific reward also may change with time. Many details of the real world applications in robotics field make it still challenging. The change of dynamics of a robot is caused by various external factors starting from temperature and ending with wear. Therefore the learning process may impossible

goes to fully convergence [3]. Frequently, the environment is dynamic so its settings cannot still the same during an earlier period of learning. External factors are often unknown. For example, light conditions have an impact on the vision system performance and the task's performance. Furthermore, the methods have to deal with uncertainties and noise of measurement and the inability to percept all states from sensing devices. When the sensors of the decision maker give noisy or incomplete information about the environment state [4], traditional Reinforcement Learning becomes difficult. The path planning algorithms are used to find optimal path from the start state to the target state with obstacles avoidance.

In many real-life environments, it is impossible for the robot to have full perception of the environment situation [2]. Unfortunately, full observability is very important for learning methods based on Marcov Decision Processes (MDPs). The agent observes the environment but these different observations or information may be given to one state because of the noisy sensor.

This problem may result from system dynamics uncertainty and system dynamics variant in first side and the inability of any model to fit these dynamics in other side [3]. Model free reinforcement learning approaches require designing behavior that is able to deal with this problem without modeling the dynamics by selecting proper reward function. Fig 1. Shows the model of agent-environment interaction in difficult conditions. This paper introduces hierarchical Gaussian Q-learning approach that has given good results in uncertain and dynamic environments. Its ability to deal with continuous actions and states gives it the possibility to implement in real life applications.

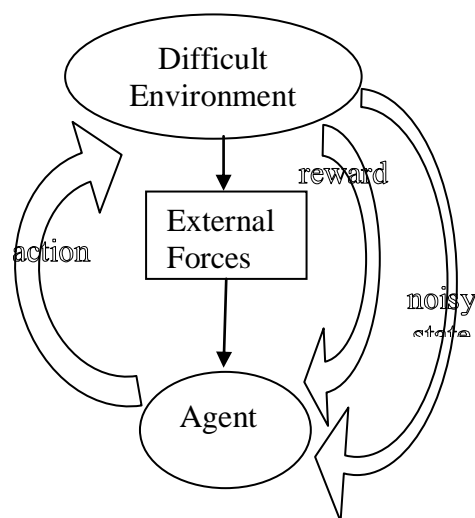


Fig. 1 the model of agent-environment interaction in difficult conditions

The paper is organized as follows: In Section 2. We review several existing approaches in hierarchicalRL, Gaussian RL and uncertain environment. Section 3explains the details of proposed algorithm: Hierarchical Gaussian Q-learning.

Section 4, concludes some of the simulation results for robot working in difficult environment. In section 5, preliminary results of Hierarchical Q-learning in 3D environment are presented. In section 6. The extension of this work in the future. Finally, the paper ends with conclusion in section 7.

2. RELATED WORK:

2.1. Hierarchical RL:

HRL decomposes the complex learning problem into small pieces to be solved easily [5]. HRL can increase the efficiency [6]. It deals with the exploration and exploitation problem. The main steps of hierarchical learning are [5]:

- 1) Automation of skills hierarchy design,
- 2) Learning of low level skills to select basic actions,
- 3) Learning of high level skills that coordinate basic skills.

One of the hierarchical RL approaches is MAXQ decomposition. It divides the Markov decision process (MDP) target into a hierarchy of small MDPs [7]. It proves convergence with probability of 1.

Hierarchical path planning approach, which depends on RL and A* with two layers structure, is presented in [8]. A* search algorithm is used in the first layer to find a geometric path to choose many samples as sub-goals for planning. In the second layer, a local path planning approach called Least Square Policy Iteration is done to plan a kinematically possible path with sub-goals. Planner can generalize well, after learning the local path.

Cooperation of multiple robots in unknown complex environments is presented in [9]. A combination of hierarchical RL and designed cooperation strategy is used. To guarantee the Real time requirement, the approach doesn't need a model of environment or trial and error to find the parameters, but they are set automatically during learning. Hierarchy consists of traditional MAXQ algorithm and segmental options in new tasks, the designed cooperation method can adapt with new task to complete it effectively.

2.2. Gaussian process in RL:

Gaussian process regression models for RL is used in continuous state space [10]. The model of Gaussian process evaluates the value function. The model is presented as functions distributions to have distributions over values in future instead of using just expectation. The Gaussian model is used for two objectives: 1) modeling the system dynamics. 2) The value Gaussian process for the value function. Uncertainties come from dynamics Gaussian process are taken into consideration during computation of the values. The value function is solved directly to avoid slow policy iterations.

2.3. Uncertain and dynamic environment:

Autonomous Underwater Vehicle has complex nonlinear dynamics [11]. The fluid dynamical forces and moments cannot be predicted easily because of their sophisticated characteristics. Above that, the environment has unknown obstacles

such as rocks and ship wrecks [11]. Moreover, the speed of water current often exceeds the speed of AUV.

Tasks of AUV in difficult environment:

AUV which is equipped with imaging sensor is used to classify objects. The classification is modeled as POMDP [12]. The objective is to focus on dynamically the next aspect to classify more quickly. Another AUV task is to look for hydrothermal vents that are existing on the sea floor with noisy information of temperature and chemical sensors [13].

AUV works in 3D environment. GPS is not working underwater. AUV needs to get information about its position. Most of sensors such as camera give noisy data. Above that, water currents and the thrusters don't always give accurate actions. Usually AUV uses INS (Inertial Navigation System) to calculate positions, but it needs to go to the surface from time to time to correct and adjust its position by using GPS. Another possible way to determine the position is to use USBL (Ultra short baseline). This way has high cost and needs another platform like ship to have connection with AUV. In the proposed method, there is no need to get accurate position of AUV, Therefore using INS is enough.

3. Proposed algorithm

3.1. Hierarchical Q-learning:

In the upper level of hierarchy, Q-learning method is used to divide the environment to multiple blocks. The number of blocks is determined by good choice of reward function in upper level.

In proposed algorithm, the reward function is;

$$r = a \cdot h_1 + b \cdot (h_2 + h_3).$$

a, b are constants.

h_1 is sum of rewards in block.

h_2 is variance of rewards in the first dimension.

h_3 is variance of rewards in the second dimension.

That robot at beginning doesn't know the accurate positions of obstacles, but it can detect them and know their scattering. Therefore, the reward function in upper level doesn't depend on the distance between robot and obstacles or between obstacles and target. Bad choice of reward function in upper level often leads to divergence. Increasing the number of blocks in upper level results in decreasing the time convergence. The process of increasing the number of blocks is still not an easy.

Fig 1. Shows the difference between flat and hierarchical RL. We can notice that flat RL needs about three times more time to converge than hierarchical RL. The time convergence (time needed to exploration and visiting states) depends on number of blocks in upper level. Increasing the number of blocks in upper level results in decreasing the time convergence. But the process of increasing the number of blocks is not an easy task and depends on reward function in upper level.

3.2. Hierarchical Gaussian Q-learning algorithm (HGQ-learning):

HGQ-learning Algorithm has following steps:

- 1) Initializations:
 - Divide the environment into number of clusters. The number of clusters differs according to the size of environment, variance of obstacles positions and computational complexity.
 - Determine the variances and means for Gaussian functions and the number of paths.
- 2) Start exploring the environment by doing random actions and taking sensors observations into consideration to know the initial (uncertain) positions of obstacles. This step is required to know the sum and scattering of obstacles inside each block.
- 3) Apply HRL to find optimal policy in upper level of hierarchy.
- 4) In lower level of hierarchy, Link each obstacle to Gaussian function which its mean is obstacle's cluster center and variance is determined according to cluster size.
- 5) The robot starts from random initial position and gets noisy data about its position which is modelled as Gaussian (mean is center of cluster that robot is in and specific variance).
- 6) Use Gaussian Q-learning to find optimal set of paths.
- 7) The robot has continuous actions and observes its state by assigning state to one of nearest Gaussian function. So it moves after time from first Gaussian function to next one that represents probabilities of next states.
- 8) While the robot follows one of pre calculated paths, it may observe new obstacle (dynamic environment), so it should change the current path to another one from the pre calculated set of paths in step 6 to avoid the obstacles. The set contains number of paths (for example 10). Every time the robot detects new obstacle, the current path is deleted and replaced with new one.
- 9) Each path is represented with samples or points which are stores to be used later, this storing is useful in applications that finding new samples costs higher. The robot should search in the set for another path that is far from detected obstacle.
- 10) Every movement from path to path results in new policy. This new policy should be added to policy dataset. This dataset is updated every time new obstacle appears.
- 11) After adding the new policy, the negative reward will be assigned to new obstacle to avoid it.
- 12) Kaman filter can be added to track the path results from policy in noisy and uncertain observations.

Mathematical equations of algorithm:

$$Q_n(s, a) = Q_{n-1}(s, a) + \alpha(r(s, a) + \gamma \max_a Q_{n-1}(s, a) - Q_{n-1}(s', a')) \quad (1)$$

$$b(\mu_i, \Sigma) = N(\mu_i, \Sigma) = B_{\max} \cdot e^{-((x-\mu_{xi})^2 + (y-\mu_{yi})^2)/2\Sigma} \quad (2)$$

$$r(\mu_i, \Sigma) = N(\mu_i, \Sigma) = R_{\max} \cdot e^{-((x-\mu_{xi})^2 + (y-\mu_{yi})^2)/2\Sigma} \quad (3)$$

$$a = A_{\max} \text{ Action is deterministic in lowerlevel of hierarchy.} \quad (4)$$

$$Q(\mu_i, \Sigma) = N(\mu_i, \Sigma) = Q_{\max} \cdot e^{-((x-\mu_{xi})^2 + (y-\mu_{yi})^2)/2\Sigma} \quad (5)$$

$$\mu_i = \begin{pmatrix} \mu_{ix} \\ \mu_{iy} \end{pmatrix} \quad (6)$$

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \quad (7)$$

$$Q_n(\mu_i, \Sigma) = Q_{n-1}(\mu_i, \Sigma) + \alpha(r(\mu_i, \Sigma) + \gamma \max_a Q(\mu'_i, \Sigma') - Q_{n-1}(\mu_i, \Sigma)) \quad (8)$$

$$\pi(x) = \arg \max Q(\mu_i, \Sigma) \quad (9)$$

$Q_n(s, a)$ is state action value function

$R(s, a)$ is reward function.

α is learning rate.

γ is discount factor.

μ_i is mean (center of one cluster in 2D environment-mean of Gaussian function).

Σ is the variance of Gaussian pdf. the large variance is more attractive for exploration than the small one.

$i=1, 2, \dots, M$, M is number of clusters.

x, y are continuous variable in state space.

$r(\mu_i, \Sigma)$ is the 2D Gaussian pdf of reward.

$b(\mu_i, \Sigma)$ is the 2D Gaussian pdf of positions.

$Q(\mu_i, \Sigma)$ is the 2D Gaussian pdf of state action value function.

Equation (1) presents the state action value function of traditional (flat) Q-learning.

Equation (2) presents the Gaussian position function (probabilities distribution function) (pdf) of robot positions in 2D grid.

Equation (3) presents the Gaussian reward function or pdf of robot rewards or punishments in 2D grid.

Equation (4) presents action of robot in 1D direction (right – left – up-down).

Equation (5) presents the Gaussian state action function or pdf of robot infinite accumulated reward in 2D grid.

Equation (6) presents the mean of cluster where the robot is in in 2D grid.

Equation (7) presents the variance of cluster where the robot is in 2D grid.

Equation (8) presents the Gaussian Q-learning in 2D grid.
Equation (9) presents the optimal policy in 2D grid.

4. Simulation results:

To confirm the efficiency of the proposed algorithm, a robot navigation in 2D environment is simulated using Matlab program. The environment has scattered fixed and dynamic obstacles. The robot should find its optimal path with the least number of steps.

HGQ-learning parameters are selected as follows:

Discount factor $\gamma = 0.5$, learning rate $\alpha = 0.7$, variance $\sigma^2 = 0.1$ (assumption: same for all Gaussian functions).

Before start learning, the environment is partially unknown. Therefore the agent should do exploration with random actions to discover the obstacles in the environment. Agent has four actions (up, down, left and right). By using sensors with noisy information, the robot can determine uncertainly the positions of obstacles in the environment. These information is not accurate but useful in upper level of hierarchical RL. After calculating the proper blocks (finding optimal policy in upper level), the robot start from the first block to the next one. When robot approaches each block, it can detect obstacles better, but uncertainty is still available. Then, robot calculates the lower level of HQ-learning to find optimal policy inside the block. Then it moves to next block and finds the next optimal policy. The process continuous until arriving to target block.

Every movement, the robot gets reward. The reward is 100 in target state, -100 in obstacles states and zero in other states.

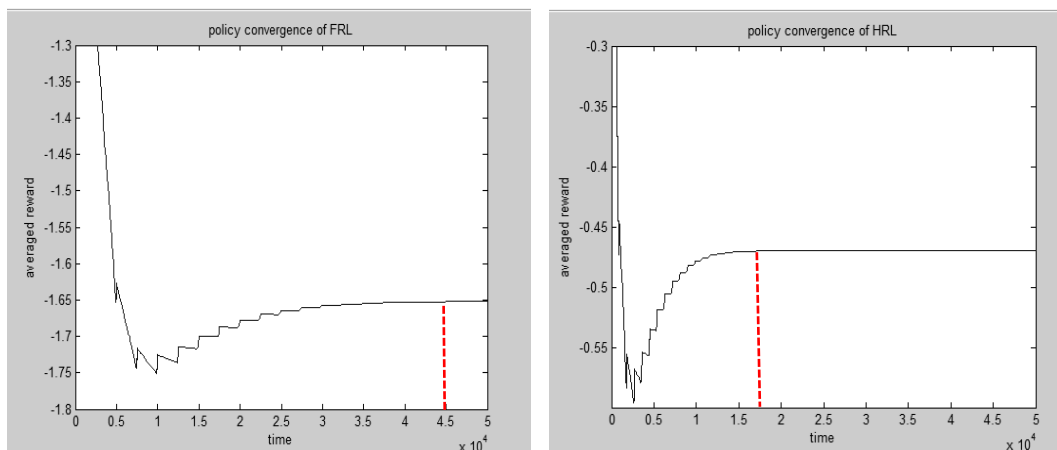


Fig 1. The convergence difference between flat (FRL) and hierarchical (HRL)

Fig 2. Shows the result of hierarchical Gaussian Q-learning in 2D environment.

Fig 2.a . Shows the upper level of Hierarchical RL, "o" indicates the agent's initial block, "*" indicates the target block. The environment is divided to 3*3 blocks.

Fig 2.b . Shows the lower level of HQ-learning.

Fig 2.c . Shows the one of the actual path of robot, the uncertainties of sensors and actions are clear.

Fig 2.d . Shows the result of Gaussian Q-learning, the result is a set of optimal paths (for example 10).

Fig 2.e . Shows the result of actual paths of robot. First it follows yellow path. But when it meets a new obstacle, it moves to new path (green one). It continuous in new path until having another new obstacle. Then it moves again to another path (cyan one). The figure shows three new obstacles.

The red path is the final optimal path that robot follows with obstacle avoidance. Fig 2.f. shows this optimal path with three obstacles.

Fig 3. Shows the representation of Gaussian Q value function, the goal at state (15,15) with high positive amplitude and obstacles with negative amplitudes.

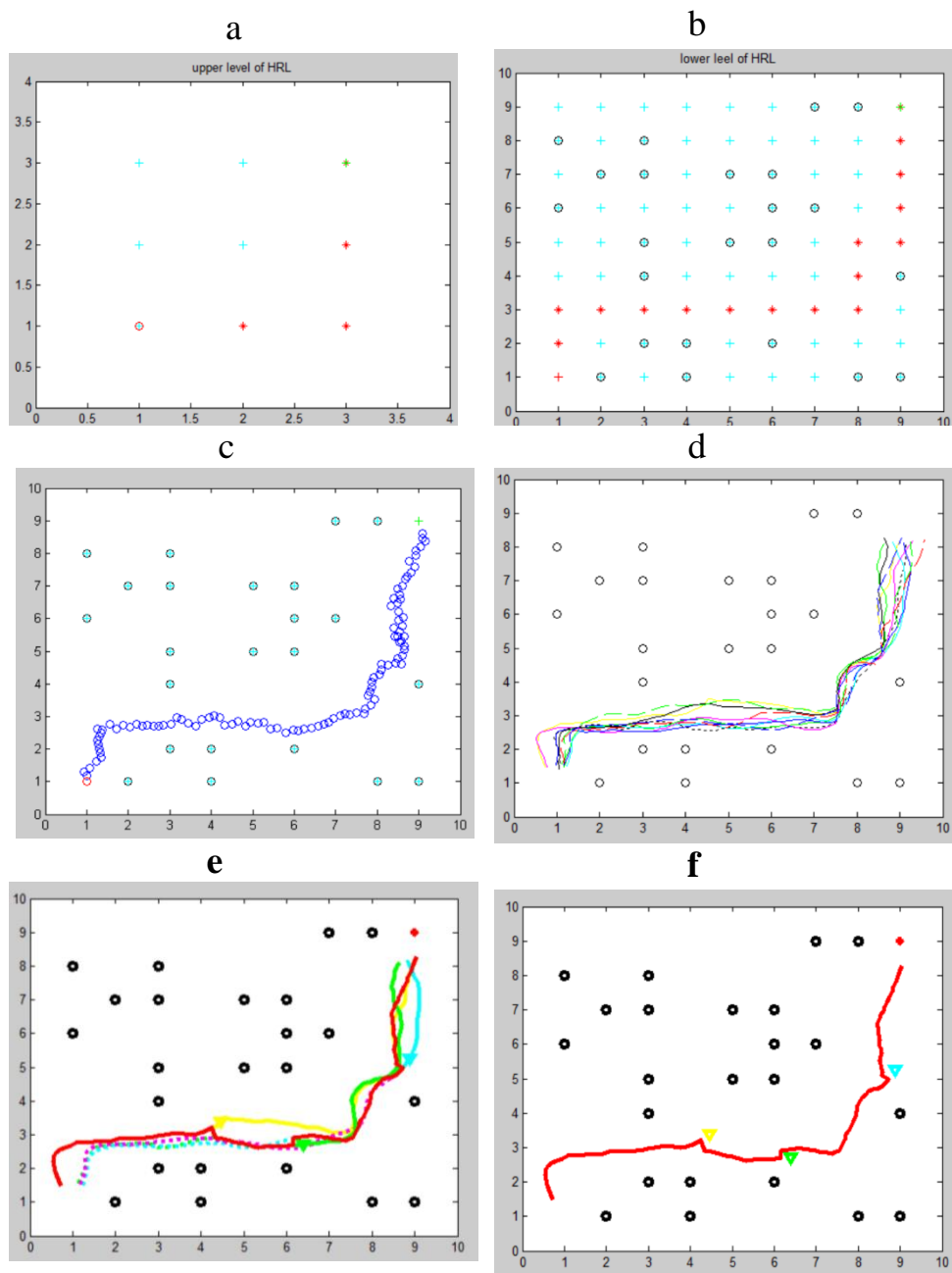


Fig 2. The result of Hierarchical Gaussian Q-learning in 2D environment.

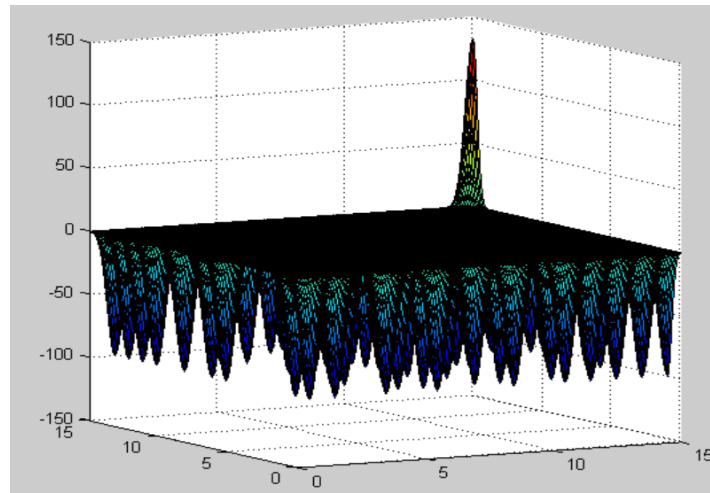


Fig 3. Representation of Gaussian Q value function

5. Hierarchical Q-learning in 3D environment:

The extension of this work is to apply the algorithm in 3D environment. Some robots like Autonomous Underwater vehicle and Unmanned Aerial vehicle perform their task in 3D environment.

Fig 4. Shows the result of applying Q-learning to find optimal policy for non-holonomic AUV in

3D environment with $10 \times 10 \times 10$ states. AUV with 4 degree of freedom (surge-sway-heave and yaw) is simulated. AUV can perform 10 actions (up-down-right-left-forward-backward-rightforward-leftforward-rightbackward-leftbackward). In Fig 3.a. there are 300 obstacles. In Fig 3.b there are 800 obstacles.

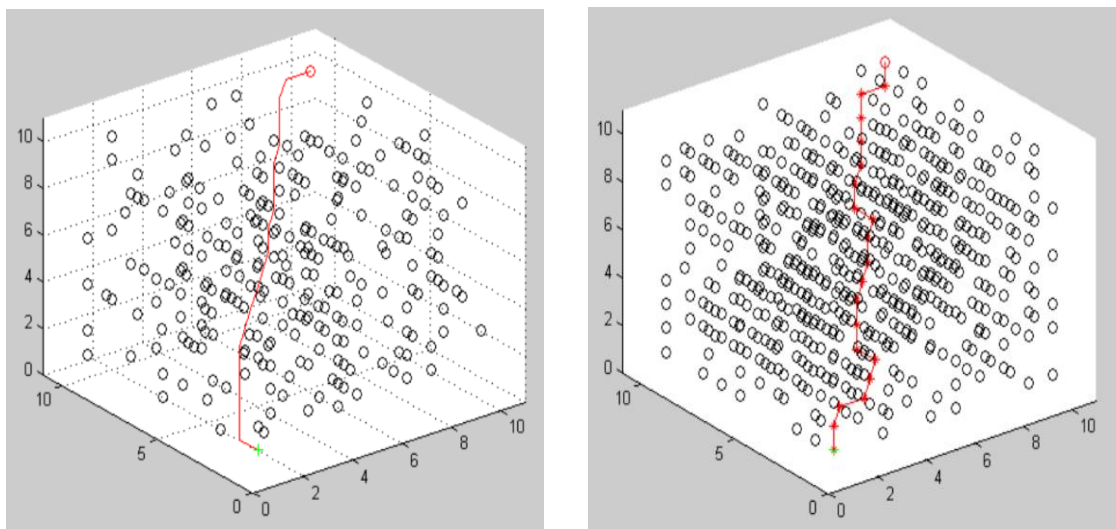


Fig 4. Path planning in in 3d environment.

6. Future work:

This work is the preliminary results of our research in using reinforcement learning techniques in uncertain environment.

In our future plan, we intend to extend the algorithm to Hierarchical multi model Gaussian Q-learning.

7. Conclusions

In this paper, hierarchical Gaussian Q-learning algorithm is proposed in partially unknown and unstructured environment. The paper discusses uncertainties that result from the actuators and sensing devices of robots and an ability of robot to adapt with external factors that affects its task such as wind or water currents. The path planning, based on combination of hierarchical reinforcement learning and Gaussian positions, rewards and actions, is required to adapt with difficult environment. The robot can work with continuous states and actions in predefined clusters. Matlab program is used to simulate the algorithm and confirms its performance.

References

- [1] Haoyu Bai, D. H. (2013). Planning How to Learn. *2013 IEEE International Conference on Robotics and Automation (ICRA)*.
- [2] Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*.
- [3] Kober, J., & Peters, J. (2012). Reinforcement Learning in Robotics A Survey. *The International Journal of Robotics Research*.
- [4] Poupart, P., & Vlassis, N. (2008). Model-based Bayesian Reinforcement Learning in Partially Observable Domains. *2008 ISAIM (International Symposium on Artificial Intelligence and Mathematics)*.
- [5] Lin, L.-J. (1993). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, School of Computer Science.
- [6] Mahajan, S. (2014). Hierarchical Reinforcement Learning in Complex Learning Problems: A Survey. *International Journal of Computer Science and Engineering*.
- [7] Thomas G. Dietterich. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*.
- [8] Guo, Q., Zuo, L., Rui Zheng, & Xu, X. (2013). A Hierarchical Path Planning Approach Based on Reinforcement Learning for Mobile Robots. *Intelligence Science and Big Data Engineering*.
- [9] Cai, Y., Yang, S. X., & Xu, X. (2013). A Combined Hierarchical Reinforcement Learning Based Approach For Multi-robot Cooperative Target Searching in Complex Unknown Environments. *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*

- [10] Rasmussen, C. E., & Kuss, M. (2003). Gaussian Processes in Reinforcement Learning. *Advances in Neural Information Processing Systems 16 (NIPS)*.
- [11] KAWANO, H., & URA, T. (2002). Motion Planning Algorithm for Non-Holonomic Autonomous Underwater Vehicle in Disturbance using Reinforcement Learning and Teaching Method. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*.
- [12] Myers, V., Defence R&D Canada - Atlantic, D. N., & Williams, D. (2011). Adaptive Multiview Target Classification in Synthetic Aperture Sonar Images Using a Partially Observable Markov Decision Process. *Oceanic Engineering, IEEE Journal of (Volume:37, Issue: 1)*.
- [13] Dearden, R., Saigol, Z. A., Wyatt, J. L., & Murton, B. J. (2007). Planning for AUVs: Dealing with a Continuous Partially-Observable Environment. *17th International Conference on Automated Planning & Scheduling*.