The 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2014)
Nov. 12 – 15, 2014 at Double Tree Hotel by Hilton, Kuala Lumpur, Malaysia

Learning Manipulative Skills Using a POMDP Framework

Ferdian Pratama, Sungmoon Jeong, Nak Young Chong

School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan
(Tel : +81-761-51-1248; E-mail: {ferdian, jeongsm, nakyoung}@jaist.ac.jp)

Abstract - Uncertainty is one of the most difficult factors to handle when we wish to develop an algorithm for robot motion planning in real circumstances. This paper presents a solution for a robot to deal with "lack of observation" in the scope of object manipulation. Considering a robotic bartender that picks up a glass filled with an unknown amount of water and tilts it to pour the water into empty glasses, the question is how to find the angle at which the giver glass is tilted to pour the water to the same level in each of empty receiver glasses. To achieve the objective, the amount of water poured is represented with mathematical models of non-linear functions, and numerical simulations are performed using the point-based value iteration algorithm for POMDP to get corresponding tilting angles of the giver glass. We found that the experimental result accuracy reaches 99.025% of similarity with the assumed mathematical model, given an initial tilting angle and the water level in the initial glass. We further verified the validity of the proposed algorithm through dynamic simulations.

Keywords - Dextrous manipulations, Learning behavior, Adaptive Systems

1. Introduction

1.1 Learning Manipulative Skills

Learning to perform certain skills is a complex process and a challenging issue in robotics, especially when latent variables exist. One of the possible causes of latent variable to appear is uncertainty. Uncertainty is one of the most difficult factors to handle when we wish to develop an algorithm for robot motion planning in real circumstances. Over the past decade, increasing attention has been paid to the imitation-based skill learning problem. It is often the case that the state of the real world is incompletely known to the robotic agent due to noisy or error-prone observations.

This paper presents a solution for the skill learning problem described using Partially Observable Markov Decision Processes (POMDP). As a sequential decision making framework with high computational cost, POMDP deals with observation uncertainty. Available for both discrete and continuous states, POMDP has become a viable, attractive solution closer to a real world representation. On the other hand, a control based approach does not work properly to solve this kind of problem, because sensors may not be available to estimate a latent variable.

Motivated by the flexibility of POMDP, we aim to use it as a solution for estimating parameters that is difficult to be estimated directly. In recent years, numerous studies have been published about the development of POMDPs in various robotics applications, including sequential decision making. Some of them are related to mobile robots development [1–6], motion planning [7–9], and solving continuous POMDPs [1, 10–13]. The available research studies focus on the computational speed, comparing their method and the state-of-the-art method. Yu [15] discussed estimation of the policy gradient in POMDP with a special class of structured policies that are finite-state controllers. She also discussed in her thesis [18] approximate solutions for POMDP and POSMDP. In another publication, Yu and Bertsekas [19] proposed a new lower approximation scheme for POMDP with discounted and average cost criterion. Brooks [1] presented a theoretical and practical result focusing on a mobile robot navigation problem in a continuous domain POMDP. He used Monte Carlo methods to estimate distributions over future parameterized beliefs, improving planning accuracy without a loss of efficiency. Zhou *et al*. [10] developed a continuous-state POMDP solver by reducing the dimensionality of the belief space via density projection and implemented it in an inventory control problem. Roy *et al*. [2] proposed a method to solve large-scale POMDPs by reducing the dimensionality of belief space using Exponential Family Principal Components Analysis [20] to represent sparse, high dimensional belief spaces using small sets of learned features.

On the contrary, our research utilizes POMDP as a solution to estimate a latent variable which is not measurable directly. To find the optimal value, we use a value iteration for a specified range of belief space. The optimal value here refers to the optimal action to be taken. After the chosen action is performed, a resulting state will occur, which is our desired estimated latent variable. In this research, we illustrate our problem of interest with a simple case study, showing that a robotic manipulator tilts a glass filled with an unknown amount of water to an appropriate angle, in order to pour the water into two empty glasses up to the same level. One of the strong points of the proposed method is that we avoid going through a pre-discretization process of dealing with the POMDP set. This means we can have a wider range set of inputs and outputs which increases the accuracy of the decision making especially in continuous parameters.

The paper is organized as follows: section 2 provides the problem statement, section 3 explains the definitions of the proposed method, and section 4 presents the experimental results. Section 5 provides discussions regarding the obtained results and finally, Section 6 concludes the paper.

2. Problem Statement

The assumptions made in this research are:

- only three identical glasses will be involved, one filled with an unknown amount of water, and the others are initially empty.
- the level of the water poured into the glass at a certain tilt angle is modeled after a non-linear equation.
- the initial tilt angle and the initial level of the water poured into the glass are known to the decision maker.

With a robotic manipulator as the decision maker, whose perception of the world is incomplete, our case study example is addressed as follows

Consider that a manipulator picks up a liquid container filled with an unknown amount of water and pours the water into identical empty containers with a certain tilt angle. With its after-pour state, how does the manipulator decide the next tilt angle in order to pour the same amount of water as the initial pouring?

We propose a solution regarding sequential decision-making process based on the point-based value iteration algorithm to generate the optimal policy, which consists of the optimal action that should be taken by the decision maker. The challenge of this research is to estimate the angle at which the glass is tilted to obtain the desired water level being poured after initially poured, considering that we could not really measure directly the water level using sensors.

3. Proposed Parameter Estimation Algorithm

The flow of pouring scenario can be divided into:

1. predetermined initial tilt angle of pouring
2. estimation of next tilt angles

During the initial pouring, the robot knows the initial tilt angle of the glass, and the resulting water level within the glass. First, the manipulator is to pour the glass of water, where the poured water is represented by exponential functions, which is obtained by trial-and-error. In this research, we define three categories of pouring processes.

Definition 1. Categories of pouring functions Three different categories were defined based on the initial water level. The 1st water poured category is defined as the function $f(x)$ valid for initial water level from 70% until 100%, with x as the pour angle, represented in the following equation

$$f(x) = 1.667(1.0615^x) \tag{1}$$

Meanwhile the 2nd and 3rd category are defined as the function $g(x)$ and $h(x)$ valid for initial water level from 40% until 70%, and 0% until 40% with the following functions:

$$g(x) = 5.9259259 \times 10^{-4}(1.14499^x) \tag{2}$$

$$h(x) = 7.8491 \times 10^{-19}(1.709975^x) \tag{3}$$

respectively.

Note that $f(x)$, $g(x)$ and $h(x)$ correspond to the relative water level function with respect to the initial water level in the grasped glass. The functions are depicted in Figure 1.

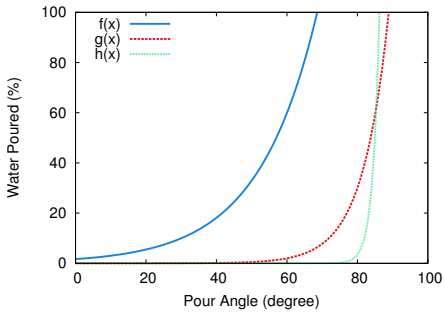


Fig. 1: Mathematical models of poured water

It should be emphasized that the POMDP set in this work was not pre-discretized, which increases the variety of inputs and output, hence allows the robot to take a more accurate action than a pre-discretized POMDP.

Definition 2. The set of states The set of states S here is represented by the water level percentage after-pour conditions in the target glass, which is defined as $S := \{s|0\% \leq s \leq 100\%\}$. This set of states corresponds to the state in every timestep, including the current and the next state.

Definition 3. The set of actions The set of actions A that is available to be taken by the robot are the pouring angle where $A := \{a|0^\circ \leq a \leq 90^\circ\}$.

Definition 4. Reward Function The reward which is assigned if the robot is doing something that is intended, is determined by the reward function of

$$R(a) = \mu_R - 2|X(s) - \mu(a)|$$

considering $\mu(a)$ is a function that maps the action to the current state $\mu(a) : a \rightarrow s$, with range of $0 \leq \mu(a) \leq 100\%$, $0 < a < 90$, and μ_R is the mean reward value represented by any positive integer, which is defined to be 5.

Definition 5. Transition function The transition function, which represents the probabilistic value of the next state that will occur, considering the action taken by the decision maker and the current state, is represented by uniform distribution in the equation

$$T(a) = \begin{cases} \dfrac{1}{a - \mu_R} & \text{if } s' < a \\ 0 & \text{else} \end{cases}$$

where s' corresponds to the after poured state.

From the known parameters, water level and pour angle, we can determine the percentage of water poured from the respective equations mentioned. As we get the result, considering that the after-pour condition is not changing, our main objective is to pour with a proper

angle to have the same amount of water being poured initially. To deal with that, we performed a point-based value iteration algorithm to obtain an optimal action to take.

Definition 6. Observation function The observation function allows mapping the obtained observations from the sensors, with respect to the actions. In this case, it is assumed to have similar expression with the transition function. Also to simplify the problem, $\forall s' \in S$, the probability mass function is identical for each action.

The main algorithm of typical POMDP case consists of two parts, the sampling of reachable belief states and the iteration process to determine the optimal action, which is the appropriate tilt angle to produce the same water level percentage with respect to the initial pouring. Algorithm 1 describes the pseudocode of the belief states sampling.

Algorithm 1 Belief Sampling (M, N, ε)

$n \leftarrow 1$
$N \leftarrow number\_of\_samples$
$b \leftarrow belief\_initial$
while $n < N$ **do**
  $a\_pmf \leftarrow \text{GAUSS}(N, \sigma, poured, i)$
  $a \leftarrow \text{GENERATE\_FROM\_PMF}(a\_pmf, a\_values)$
  $s' \leftarrow \text{GENERATE\_FROM\_PMF}(trn(a), s\_values)$
  $o' \leftarrow \text{GENERATE\_FROM\_PMF}(obs(a), s\_values)$
  $b_{new} \leftarrow \text{SE}(a, b, states, trn, obs)$
  $arg1 = |b_n[0] - b_{new}|$
  if $n > 1$ **then** $arg2 = |b_n[n-1] - b_{new}|$
  else $arg2 = [\infty] \times 23$
  if $min(arg1, arg2) > \varepsilon$ **then**
    $n+ = 1$ $b_n = append(\text{b`new})$
return b_n

The next state s' and the observation o' go through the similar procedure with the action selection, but using the set of transition trn and the set of observation obs respectively, instead of the set of action a. SE represents the state estimator, which is mathematically represented by Equation 4, and the pseudocode is described in Algorithm 2, where $P(o'|a,b)$ is the normalization part.

$$\begin{aligned} SE(b,a,o') &= b'(s') \\ &= P(s'|o',a,b) \\ &= \frac{P(o'|s',a,b)P(s'|a,b)}{P(o'|a,b)} \\ &= \frac{P(o'|s',a)\sum_s P(s'|a,b,s)P(s|a,b)}{P(o'|a,b)} \\ &= \frac{P(o'|s',a)\sum_s P(s'|a,s)b(s)}{P(o'|a,b)} \end{aligned} \tag{4}$$

Then it will compare the absolute difference between b_n and the new belief state generated b_{new} and find the minimum. If the resulting value is larger than ε, the new belief will be appended to the set of sampled belief states. Parameter ε ensures that the sampled results are spread

Algorithm 2 State Estimator (A, S, b, trn, obs)

$total \leftarrow 0$
for all $s \in S$ **do**
  for all $x, y \in (trn(a), b)$ **do**
    $b(s) \leftarrow x \times y$
    $belief_{new} = belief_{new} \times obs(a)$
    $total+ = sum(belief_{new})$
return $b_{new}/total$

out in belief space, which is set to be 0.001. The returned value b_n, is the sampled belief states which will be used as an input in the value iteration procedure in Algorithm 3.

Algorithm 3 Value iteration ($\mathscr{M}$, b_n, N, γ)

$\alpha_{max} \leftarrow -\infty$
for $n \leftarrow 0 \rightarrow N$ **do**
  for all $s \in S$ **do**
    for all $a \in A$ **do**
      for all $o \in O$ **do**
        $b_{new} \leftarrow \text{SE}((a), b_n[n], (s), trn, obs)$
        $\alpha \leftarrow sum(trn(a) \times obs(a) \times rwd(a) \times b_{new})$
        if $\alpha > \alpha_{max}$ **then**
          $\alpha_{max} = \alpha$
        $value(a) \leftarrow rwd(a) + \gamma \times \alpha_{max}$
FIND_ACTION(b`n, value, a)

For the 2nd step, the value iteration process is described in Algorithm 3. It would continue to loop as many as the amount of sampled belief state N, and performs nested loop for each states, actions, and observations, then updates the new belief b_{new} using the state estimator procedure described in Algorithm 2. Then the future values α is calculated, and find the maximum value of α during the nested loops. It proceeds to the value calculations and moves on to determining the optimal action, described in Algorithm 4.

Algorithm 4 Find optimal action $(b_n, a, value)$

$prev\_action\_value \leftarrow 0$
$prev\_action\_taken \leftarrow'$ *none*$'$
for $a \in A$ **do**
  $action\_value \leftarrow sum(b_n \times value(a))$
  $taken\_action \leftarrow a$
  if $action\_value > prev\_action\_value$ **then**
    $prev\_taken\_action \leftarrow taken\_action$
return prev_taken_action

In determining the optimal action to be taken, just simply find the maximum value of actions, and select the action as the optimal action.

4. Experimental Procedure & Results

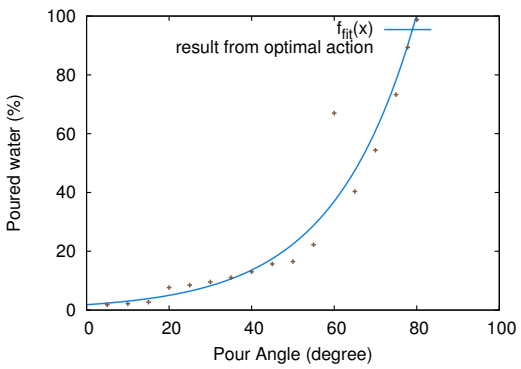
The experiment procedures are listed as follows:
1. initial pouring with a certain water level and tilt angle
2. obtain the percentage water level remained within the grasped glass

3. estimate the optimal tilt angle based on the remaining water level
4. repeat for other initial tilt angles θ for $0 < \theta < 90$

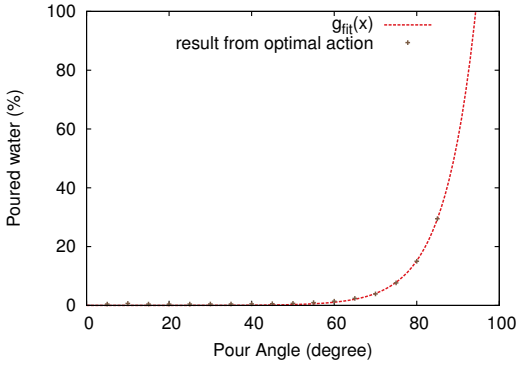
The experiment were tested for all three categories of pouring process, randomly selected initial water level, in this case 70%, 45%, and 30% which represent their respective categories. The experiment consists of two parts, numerical and dynamics simulation.

Numerical Simulation The proposed method was simulated numerically in Python. The graphs in Figure 2 depict the optimal action data taken for various angles, being fitted to each curve from their respective categories. The curves are fitted based on the implementation of non-linear least-squares (NLLS) Levenberg-Marquardt Algorithm as shown in Table 1. Coefficients a and b correspond to the basic exponential equation $a(b^x)$. Finally, we can obtain the final value of a and b for each equation's category, which are used to plot the graph in Figure 2.

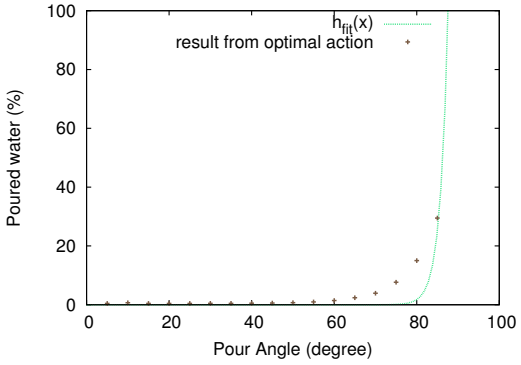
Dynamics Simulation As an illustration in robotics applications, an experiment using Open Dynamics Engine (ODE) available within the V-REP robot simulator was performed to show one of the results from the above-mentioned results. The setup is defined as follows: a Mitsubishi PA-10 manipulator equipped with Barrett Hand as the grasper, grasps a glass filled with a certain amount of water which is known by the manipulator, pours to an empty glass with a certain initial tilt angle, and pours to the other identical empty glass with the optimal tilt angle to reproduce the same amount of poured water level as the initial pouring. The pouring process is illustrated in Figure 3b and Figure 3c. In this case, the pouring position for each glass are pre-determined without affecting the decision making, hence the only challenge is how to obtain similar water level with a proper action, which is the optimal tilt angle. The water particles are simulated using a frictionless tiny spheres with density set to be $1000kg/m^3$, which is water density. Acceleration of pouring were not taken into account since the pouring velocity is set to be constant. The pouring results which compare the both water level are shown in Figure 3d.



(a) data fit $f(x)$ Curve with 70% water level



(b) data fit $g(x)$ Curve with 45% water level



(c) data fit $h(x)$ Curve with 30% water level

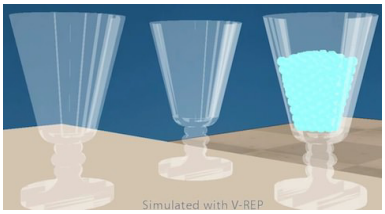
Fig. 2: Curve fitting optimal action data with the respective function from experimental results

5. Discussion

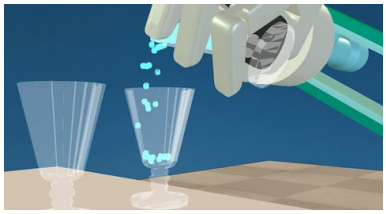
Numerical Simulation Analysis Since we assumed to have a mathematical model as the pouring function in the first place, it is more convenient for us to check the accuracy of our estimation algorithm by the difference of the next glass tilt angles with the initial tilt angle. The results of the data fitted with $f(x)$ curve show the equation obtained $f_{fit}(x) = 1.84406(1.05125^x)$. If we compare with the assumed mathematical equation $f(x) = 1.667(1.0615^x)$, we can calculate each coeffiecient's percentage error by calculating the absolute difference with respect to the initial equation, which we obtained 9.60% for coefficient a and 0.975% for b. The error for both coefficients of the 2nd category for $f(x) = 5.9259259 \times 10^{-4}(1.14499^x)$ and $f_{fit}(x) = 0.00039676(1.141^x)$ appeared to be $Err\_a_{45} = 33.05\%$ and $Err\_b_{45} = 0.348\%$. For the 3rd category, the errors obtained for $f(x) = 7.8491 \times 10^{-19}(1.709975^x)$ and $f_{fit}(x) = 5.94028 \times 10^{-19}(1.70166^x)$ are $Err\_a_{30} = 2.431 \times 10^{-37}\%$ which almost reaches 0 technically, and $Err\_b_{30} = 0.486\%$. Since the parameter a is the y-intercept of the graph, meaning the graph crosses the y-axis at the point $(0,a)$, we can just focus on the difference of parameter b, which in this case since $b > 0$ determines the growth rate of the function. Also we desired less percentage error of coefficient b. Summarizing the percentage errors of b, which are $Err\_b_{70} = 0.975\%$, $Err\_b_{45} = 0.348\%$, and $Err\_b_{30} = 0.486\%$. Hence, we can determine that the most similar growth rate exponential function with our initial assumption of exponential

Table 1: Curve Fitting Result

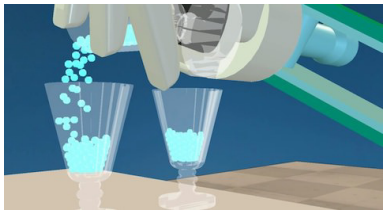
		Coefficient		Asymptotic Standard Error			
		Initial	Final	+/−	%	Iterations	Fitted equation
$f_{fit}(x)$	a	1	1.84406	0.6021	32.65	13	$1.84406(1.05125^x)$
	b	1	1.05125	0.004234	0.4028		
$g_{fit}(x)$	a	0.5	0.00039676	0.0001104	27.84	615	$0.00039676(1.141^x)$
	b	1	1.141	0.00366	0.3208		
$h_{fit}(x)$	a	1	5.94028×10^{-19}	6.49×10^{-20}	10.93	20866	$5.94028 \times 10^{-19}(1.70166^x)$
	b	1	1.70166	0.0002097	0.1233		



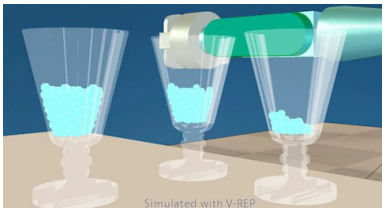
(a) Conditions of 60% initial water level



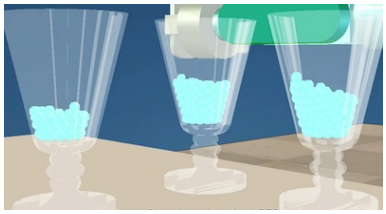
(b) First pouring with pre-determined initial angle 60°



(c) Optimal pouring result with angle 70°



(d) Comparison of the remaining water in initial glass(right), initial pouring(middle) and estimation pouring(left)



(e) Fail pouring illustration result with 63° as the output angle

Fig. 3: Consecutive pouring experiment with V-REP

function is the 2nd category, and our estimation algorithm works best with the 2nd category. Experimentally, with the assumption of exponential functions, the maximum error of the pouring estimation based on the optimal action taken reaches 0.976%, which is by the 1st category.

Dynamics Simulation Analysis We can determine the numbers of water particles initially in the simulator, but it is currently not feasible to compute the existing particles within the glass. Since there is no metric available to measure the amount of water that have been poured in dynamics simulation using V-REP, we visually compare the water level. Figure 3e illustrates the non-optimal pouring angle taken by the robot. It emphasized our strength point, which implies that the robot can have a more specific output instead of, for instance 5° discretized output and the difference of 7° from the optimal angle yield large difference in the water level. As seen in Figure 3d, even though the water surface is not flat, the proper estimation leads to similar water level on both target glasses, where the optimal action to be taken by the agent were 70° for the 2nd pouring. We will investigate the problem in a real humanoid robot as one of the future works.

Generalization The performed task of manipulative skill discussed is just one illustration of how POMDP can be use to estimate a latent variable in object manipulation. In general, the proposed principle can be analogous to another application domain as long as a latent variable is involved. In recent applied pharmaceutical research, no sensors available to determine a perfect mixing for liquid-liquid mixing. The state-of-the-art method to determine a perfect mixing is to mix the mixture manually based on consistency of mixing speed and time. Therefore the sample taken from any point of the mixture is expected to have exactly the same composition. POMDP will come in handy to solve this kind of problem, considering there is no sensing device to determine whether the particles are distributed evenly within a mixture. The robot arm stirring the mixture must consider those latent variables in order to stop stirring when perfect mixing occurs. The proposed method can be applied to a more general application domain with similar ideas on estimation of latent variables.

6. Conclusion

This paper presents a solution to estimate a latent variable which are unlikely to be observed or quantitatively measured directly using sensors. The purpose of this research is to demonstrate the use of POMDP as an estimation technique, which is a robust yet effective tool to take uncertainty into account in decision making repre-

sentation. To illustrate the estimation fidelity within manipulative skills, a water pouring experiment was chosen. The pouring process was modeled as exponential equations for different water level percentage. Numerical simulations using Python were performed to evaluate the accuracy of the proposed method and demonstrated using dynamic simulations. The results show that the minimum accuracy reached 99.025%. In future works, we will apply the proposed method to a more complex domain, aiming at achieving sophisticated robotic skill learning.

References

[1] A. M. Brooks, "Parametric POMDPs for Planning in Continuous State Spaces," Ph.D. Thesis, The University of Sydney, 2006.

[2] N. Roy, G. Gordon and S. Thrun, "Finding Approximate POMDP solutions Through Belief Compression," *Journal of Artificial Intelligence Research (JAIR)*, Volume 23, Pages 1-40, 2005.

[3] J. Pineau, and S. Thrun, "High-level robot behavior control using POMDP," *in, Proceedings of AAAI Conference on Artificial Intelligence (AAAI-2002)*, 2002.

[4] O. ASIK, and H. L. Akin (2013), "Solving Dec-POMDPs by Genetic Algorithms: Robot Soccer Case Study," *MSDM-2013 Workshop at 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Saint Paul, Minnesota, USA. to be published.

[5] B. Eker, and H. L. Akin, "Solving Decentralized POMDP Problems Using Genetic Algorithms," *Journal of Autonomous Agents and Multi-Agent Systems*, Volume 27, No.1, Pages 161-196, 2013.

[6] B. Eker, E. Ozkucur, C. Mericli, T. Mericli, and H. L. Akin, "A Finite Horizon DEC-POMDP Approach to Multi-robot Task Learning," *The 5th International Conference on Application of Information and Communication Technologies, AICT2011*, 2011.

[7] J. Van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *International Journal of Robotics Research*, 2012

[8] N. E. Du Toit, and J. W. Burdick, "Robotic Motion Planning in Dynamic, Cluttered, Uncertain Environment," *in IEEE International Conference on Robotics and Automation*, 2010

[9] J. Van den Berg, S. Patil, and R. Alterovitz, "Motion Planning under Uncertainty using Differential Dynamic Programming in Belief Space," *15th International Symposium on Robotics Research (ISRR 2011)*, 2011

[10] E. Zhou, M.C. Fu and S.I. Marcus, "Solving continuous-state POMDPs via density projection," *in, IEEE Transactions on Automatic Control*, Volume 55, No. 5, Pages 1101-1116, 2010.

[11] J. Van den Berg, S. Patil, and R. Alterovitz, "Efficient Approximate Value Iteration for Continuous Gaussian POMDPs," *in, Proceedings of AAAI Conference on Artificial Intelligence (AAAI-2012)*, 2012.

[12] M. P. Diesenroth, and J. Peters, "Solving Nonlinear Continuous State-Action-Observation POMDPs for Mechanical Systems with Gaussian Noise," *JMRL: Workshop and Conference Proceedings of European Workshop on Reinforcement Learning*, Volume 24, Pages 1-14, 2012.

[13] T. Erez, and W. D. Smart, "A scalable method for solving high-dimensional continuous POMDPs using local approximation," in Proceedings of 26th conference on uncertainty in artificial intelligence, 2010

[14] D. P. Bertsekas, "*Dynamic programming and optimal control*", Athena Scientific, 1995.

[15] H. Yu, 'A Function Approximation Approach to Estimation of Policy Gradient for POMDP with Structured Policies'. *Journal of Computing Research Repository*, 2012. CoRR [Online]. Available at: http://arxiv.org/abs/1207.1421 (Accessed: 9 March 2013)

[16] J. Baxter, and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research (JAIR)*, 2001.

[17] V. Konda, "Actor-Critic Algorithms," *Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA*, 2002.

[18] H. Yu, "Approximate Solution Methods for Partially Observable Markov and Semi-Markov Decision Processes,", Ph.D. Thesis, M.I.T., Cambridge, MA, February, 2006.

[19] H. Yu and D. P. Bertsekas, 'Discretized Approximations for POMDP with Average Cost'. *Journal of Computing Research Repository*, 2012. CoRR [Online]. Available at: http://arxiv.org/abs/1207.4154 (Accessed: 9 March 2013)

[20] M. Collins, S. Dasgupta and R. E. Schapire, "A generalization of principal component analysis to the exponential family," *Advances in Neural Information Processing Systems*, Volume 14, Pages 617-624, 2001

[21] D. Hsu, W. S. Lee and N. Rong, "A point-based POMDP planner for target tracking," *in IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008)*, Pages 2644-2650, 2008.

[22] H. Kurniawati, D. Hsu and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," *In Proceeding of Robotics: Science and Systems*, 2008.

[23] K. Murphy, "A survey of POMDP solution techniques," Technical report, U. C. Berkeley, 2000

[24] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, Volume 12, No.3, Pages 441450, 1987.

[25] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and related

stochastic optimization problems," *Journal of Artificial Intelligence*, Volume 147, No. 1-2, Pages 5-34, 2003.

[26] W. S. Lovejoy, "Computationally feasible bounds for partially observed Markov decision processes," *Operations Research*, Volume 39, No. 1, Pages 162175, 1991.

[27] M. Hauskrecht, "Value function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, Volume 13, Pages 3395, 2000.

[28] M. T. J. Spaan, and N. Vlassis, "Perseus: Randomized Point-based Value iteration for POMDPs," *Journal of Artificial Intelligence Research*, Volume 24, Pages 195-220, 2005.

[29] J. D. Williams, "Partially Observable Markov Decision Process for Spoken Dialogue Management," *PhD Thesis*, Churchill College and Cambridge University Engineering Department, The University of Cambridge, 2006.