JAIST Repository
https://dspace.jaist.ac.jp/

Title	Adaptive Point-Based Value Iteration for Continuous States POMDP in Goal-Directed Imitation Learning
Author(s)	Pratama, Ferdian Adi; Lee, Hosun; Lee, Geunho; Chong, Nak Young
Citation	2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI): 249-254
Issue Date	2012-11
Type	Conference Paper
Text version	author
URL	http://hdl.handle.net/10119/11412
Rights	This is the author's version of the work. Copyright (C) 2012 IEEE. 2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2012, 249-254. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

The 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2012)
Nov. 26-28, 2012 in Daejeon Convention Center, Daejeon, Korea

Adaptive Point-Based Value Iteration for Continuous States POMDP in Goal-Directed Imitation Learning

Ferdian Adi Pratama, Hosun Lee, Geunho Lee, and Nakyoung Chong

School of Information Science, Japan Advance Institute of Science and Technology, Ishikawa, Japan
(Tel : +81-761-51-1248; E-mail: {pura-f, hosun_LEE, geun-lee, nakyoung}@jaist.ac.jp)

Abstract - In motion planning and robot navigation, continuous domain would be the natural way of representation of state space. However, discretization is needed in order to deal with continuous state space. Results precision depends on the discretization, which leads to a problem of "curse of dimensionality". We present a new approximation approach of goal-directed imitation learning algorithm using point-based value iteration algorithm deals with continuous domain in motion planning Partially Observable Markov Decision Process with desirable precision. We demonstrate our algorithm in the V-REP robot simulator, to validate the experimental result.

Keywords - Goal-Directed Imitation, Sequential Decision Making, Motion Planning, POMDP.

1. Introduction

In general context of learning, trying to copy a movement or a certain task demonstrated physically by others is a common way for humans to accomplish similar task. In learning how to dance by imitation, human tries to match their limbs configurations to others to get the same posture. On the other hand, in case of learning how to pour a water properly from a glass to another by imitation, similar limbs configuration sometimes not a big issue, as long as the water is properly poured.

Two different kinds of imitation learning are well-known in robotics, namely behavioral-based and goal-directed based imitation learning. Behavioral-based Imitation Learning is similar with learning how to dance, which means the imitator's main objective is to get the same posture and behavior as the demonstrator. Goal-directed imitation learning only concerns about how to get the same outcome of a certain task demonstrated. Due to the fact that it concerns about the physical link configuration in an articulated robot, it has a very high complexity especially in a redundant manipulator. Goal-directed imitation learning is useful in a case where we need a manipulator to accomplish a task imitated by human or other robot. It focuses on getting things done instead of the manipulator posture. Using a goal-directed imitation learning algorithm, we don't need to program the robot explicitly to do a certain task performed by others.

To achieve a solid-structured and organized machine learning, we need a mathematical framework of learning environment for the agent (decision maker). Markov models are popular mathematical tools used widely in machine learning research. Comprises four different types, Markov Chain (MC), Hidden Markov Model (HMM), Markov Decision Process (MDP), and Partially Observable Markov Decision Process (POMDP), each of them provides different properties under particular conditions to represent the learning environment in machine learning algorithm. Since action transitions are not controllable in both markov chain and HMM, MDP and POMDP provides a controllable action transition property for learning environment representation. MDP describes a deterministic relation between an agent and the environment. Analogous to markov chain in terms of the agent's current state certainty, the agent in MDP representation knows exactly its current state. Meanwhile, POMDP is a stochastic representation of learning environment with addition of controllable action transitions, and also deals with uncertainty elements of the agent's current state.

Since a more realistic environment representation deals with uncertainty, a stochastic representation is preferable, which makes POMDP a better representation as a learning environment. Despite of the natural representation, it is intractable to get an exact POMDP solution [1]. There has been vast development of practical POMDPs algorithm [2]-[4] in the recent years. Value iteration based on the discretization of each continuous state space is the common way to solve POMDP. It is practical if the number of states and horizon is relatively small. When it comes to a large amount of states and horizon, "Curse of Dimensionality" occurs, when the computational complexity increases drastically with the dimension of belief space, and it becomes intractable to solve. This is a dilemma where a fine discretization would lead to a "Curse of Dimensionality", and poor discretization would lead to poor representation of the state space.

In this paper, we present a new approach of goal-directed imitation learning algorithm, Adaptive Point-Based Value Iteration (APBIt), based on asynchronous value iteration for sampling the state space appropriately rather than complete discretization, hence desired precision can be achieved without significant increase of the computation time. We performed numerical simulations and demonstrated our algorithm for motion planning environment using a 3D robot simulator; V-Rep [5].

2. Background

2.1 Preliminary

A Partially Observable Markov Decision Process (POMDP) set M, is a tuple $\langle S, A, T, R, Z, O \rangle$, comprises a finite set of state S, a finite set of action A, a finite

set of observation O, a transition function T, a reward function R and an observation function Z. A transition function $T : S \times A \rightarrow \Pi(S)$ represents a mapping of state and action. A reward function $R : S \times A \rightarrow \Re$ is equivalent to $R[s, a]$, a finite set of observation $O : S \times A \rightarrow \Pi(O)$ is equivalent to $Pr(o|s', a)$, and observation function $Z : S \times A$ can be interpreted as $Z(s', a, o') = Pr(O^{t+1} = o'|S^{t+1} = s', A^t = a)$.

A POMDP is a realistic version of MDP in a sequential decision making. The agent takes an action $a \in A$ from a certain state $s \in S$ to another state $s' \in S$. Real life situation deals with probabilistic owing to uncertainty. During the decision making from a certain state to another, uncertainty can be represented by a transition probability $T(s, a, s')$. Due to the incomplete information of its current state, the agent must make an observation $o \in O$. The observation itself also deals with uncertainty, and it can be represented by $Z(s, a, o) = Pr(o|s, a)$

A POMDP can be converted into a continuous-state MDP by introducing a notion of the belief state [6]. To identify its current state in a stochastic environment, the agent compute a probability distribution over states. A belief state b is a representation of agent's current state given the history of action and observation. The notation $b(s)$ represents the probability that the current state of the agents is $s \in S$.

A policy π is defined as mapping related to current state s and which action a to take. Due to uncertainty elements in POMDP, a policy π can be defined as $\pi : S \times T \mapsto A$. Reward function $R : S \times A \mapsto \mathbb{R}$ maps a numeric reward for each state and action. An immediate reward notation given the current state and action to take would be $R(s, a)$. The output of POMDP is an optimal policy tree π^* and the agent must choose actions to maximize the expected total reward. To do that, based on the current state and taken action of the agent, it receives a reward $R(s, a)$. For optimal policy, the agent takes an action that has maximum value considering the belief state in a certain state. The agent also have to consider the future reward as well as immediate reward. What we need is the sequence of actions that have the maximum expected reward instead of individual actions. An agent can measure the reward value being in a certain state by means of value function. We can represent the optimal value function in a t-step policy tree $V(b, a)$ as:

$$V(b, a) = \sum_{s \in S} b(s)R(s, a) + \gamma \sum_{o \in O} Pr(o|a, b)V^*_{t-1}(b^a_o)$$

where b^a_o is the belief state from b after taking action a and receiving observation o. Parameter γ is specified to be geometric discounting factor, where $\gamma < 1$.

2.2 Related Work

In the real world POMDP problems, it is unnecessary to compute the optimal policy for the full belief space and considered more practical and easier to compute a good approximation of the optimal value function [8]. Recently quite a lot of algorithms for solving POMDP approximately are available. Thrun [9] addressed an approximate approach deals with continuous-state and action POMDPs using MonteCarlo Method to sample belief representation and propagation. Brooks [10] presented a theoretical and practical results focuses on a mobile robot navigation problem in a continuous domain POMDP. He used MonteCarlo methods to estimate distributions over future parametrised beliefs, improving planning accuracy without a loss of efficiency.

Kurniawati *et al.* [4] emphasized on the exploitations of computational efficiency improvements from the notion of optimally reachable belief spaces. Their idea is to sample a subset of belief points reachable from initial belief point $b_0 \in B$, which expected to reduce the number of samples. Zhou *et al.* [11] developed a continuous-state POMDP solver method by reducing the dimensionality of the belief space via density projection and implemented it in an inventory control problem. Particle filtering, which is one of a MonteCarlo Method, which needs a large number of samples, is not a good solution to obtain a reasonable approximation of belief state. Hence, they incorporate the notion *density projection* into the particle filter, by approximating the projected belief space by a certain density.

Roy *et al.* [8] proposed a method to solve large-scale POMDPs by reducing the dimensionality of belief space using Exponential family Principal Components Analysis [12] to represent sparse, high dimensional belief spaces using small sets of learned features. Their idea is to plan in this low-dimensional space, so they can determine the POMDP model policy which have higher order of magnitude that the model which can be handled by conventional techniques. They also demonstrated their algorithm on a synthetic problem and on a mobile robot navigation task.

For some of the paper described, MonteCarlo method is used to optimize and improve the performance. Motivated by the presently available approximate POMDP algorithm and due to the scarcity of imitation learning algorithm, we would like to give a contribution by developing a relatively easy-to-implement imitation learning algorithm and relating imitation learning using POMDP as the environment representation under several limitations.

3. Problem Statement

We categorized a goal-directed imitation learning further into two: learning to imitate something and learning to do something properly. The former lies in more general context, which means even if the task performed by the demonstrator is not the correct way to do something, the learning agent will try to imitate it to get similar results compared with the results during it was demonstrated. The latter, however, must conform a crucial condition: the task performed has to be the proper or correct way of doing something. The latter, which lies in a more specific context, will come in very handy in a situation where we need to accomplish the same task performed in a different environment. The changed envi-

ronment makes it difficult to accomplish if the imitation is performed using the former category. Meanwhile, the latter category has a capability to adapt the decision making process with the current environment, while trying to maintain a proper performance of the demonstrated task.

This research deals with the former category, since it is a more general context and can be extended later into the latter one. The task demonstrated is to be assumed as pouring a water from a glass to another, and it will involve only a motion planning.

3.1 Problem Definition

We formally address the goal-directed imitation learning using POMDP as the learning environment representation as follows:
Assumed a glass of water was poured by the demonstrator to an empty glass, how to perform an imitation resulting a similar end-effector cartesian coordinate position to the demonstrator?

The goal-directed imitation learning problem above can be decomposed into two related subproblems:
- **Sub-Problem 1**, *Imitation Learning* How to achieve a sufficiently-good goal-directed imitation learning in motion planning?
- **Sub-Problem 2**, *Decision Making* How to perform sequential decision making resulting similar outcome with inputs acquired by sensors surrounding the learning environment?

The two sub-problems relates to each other. If we solve the question in sub-problem 2, consequently we will solve the sub-problem 1.

4. Algorithm Description

4.1 Imitation Learning Representation using POMDP

To recognize what is going on in the surrounding, a perception system is needed, while action is related to actuators or a mechanism which generates force or physical movement. Treating both elements as an individual system will not be very useful, compared if both elements treated as a one bigger system. We need an intelligent system, bridges the perception and action. Three basic elements shown in Figure 1: perception, action, and the one that connects both elements, an intelligent system.

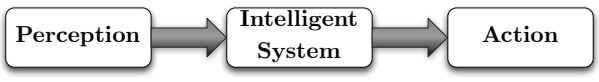


Fig. 1 The relation of intelligent system with perception and action

In this research, we are trying to connect the perception and action with imitation learning representing the intelligent system, by choosing POMDP as the mathematical framework. Vision system will represent the perceptionand motion generation will represent the actionshown in Figure 2.

POMDP acts as a sequential decision making tool through a stochastic environment representation, processing the data acquired by the vision system, and its output

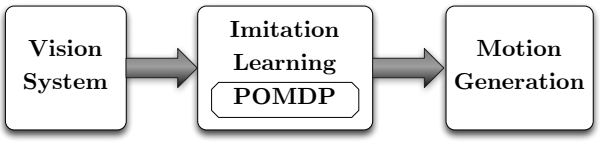


Fig. 2 POMDP as the environment representation in imitation learning

will be delivered to the actuatorto accomplish the task demonstrated.

4.2 Reformulation of Learning Algorithm

Learning agent is formally defined as the decision maker, and demonstrator is the one which performed a certain task to be imitated by the agent. A set of horizon available for decision making H_n as the *n-number of horizons available* is denoted as $\{h_0, h_1, ..., h_{n-1}\}$. Dealing with continuous states POMDP, we specify the $\{s_{min}, s_{max}\}$ range of states denoted by S_i as the ith horizon. The $\{s_{min}, s_{max}\}$ range for each horizon are possible to have different value. We need to have a boundary to perform uniformly distributed sampling.

Definition 1. (SAMPLING BOUNDARY) *A boundary for uniformly distributed random sampling denoted as* $\{0, h_{min}\}$*, and* h_{min} *which represents the maximum boundary is defined as*

$$h_{min} := \min_{h \in H} [s_{max} - s_{min}]$$

Definition 2. (UNIFORMLY DISTRIBUTED RANDOM NUMBER SAMPLING) *Given the sampling boundary* $\{0, h_{min}\}$*, a set of uniformly distributed samples can be obtained by performing a uniformly distributed random number sampling, defined as a set of predefined n-number of samples* $\{u_0, u_1, u_2, ..., u_{n-1}\}$ *denoted by*

$$U_n := \{u_0, u_1, u_2, ..., u_{n-1}\},$$

where U_n *represents the set of n-number of samples.*

The set of samples U_n will act as the substitute of continuous states, which makes it straightforward for us to perform a point-based value iteration based on the obtained samples. The notation of states and action onwards will refer to the samples obtained.

The imitation learning algorithm inputs are specified to be the position captured by the vision systems after the system gets a trigger that a certain movement is ready to be imitated. The inputs needed by the imitation learning algorithm are categorized into two: main input I_m and observer inputs I_o^n.

Definition 3. (INPUT DESCRIPTION) *Given the main and observer vision systems, main input* I_m *is defined as a set of two-valued input contains the position range approximated by each vision systems. Observer input* I_o^n *holds the same definition with main input, and can be interpreted as* n*th observer vision system. Both input formulations are generalized as* I*, denoted as*

$$I := \{P_{min}, P_{max}\}$$

where P as the approximated position.

A single main vision system is described as the vision system that monitors the demonstrator directly. Observer vision systems are described as systems that monitors the whole learning environment, including the range $\{s_{min}, s_{max}\}$. Gaussian function is employed as the reward, belief and observation function. As the reward function, it is assumed that the uncertain peak value with respect to a certain sample V_p has the same percentage of getting errors within a certain range, due to the symmetrical characteristics of the function. Each of the learning algorithm inputs are described as a set consists of two elements describing the approximate position predicted by the vision system. Therefore, a position approximator algorithm, which supposed to be integrated in the vision system, is needed to produce the specified vision system outputs. Due to out of the scope of this paper, the inputs are simply assumed. The two-valued input set determines the peak of the gaussian function which will determine the whole function configuration.

Definition 4. (BELIEF FUNCTION) *Consider observer sensors covering all the continuous space, belief function, affected by observation as parameter, values of samples, and horizon, is defined as gaussian probability density function which standard deviation and mean depends on the two-valued inputs, and denoted as*

$$b(u_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(u_i-\mu)^2}{2\sigma^2}}$$

where μ as the mean, σ^2 as variance, σ as the standard deviation, and u_i as the samples value.

Belief function affects value function $V(b, u_i)$. Reward function, which also defined as gaussian probability density function, affects current and next horizon value. Having the same formulation with belief function, observation function, also defined as gaussian probability density function, affects the only the next horizon value. Observation function lies within next horizon value $alpha'$, determining the iteration sum of each next horizon sampled value.

Definition 5. (BACKUP FUNCTION) *Consider taking all the possible values in the next horizon into the current horizon, the backup function $\alpha_{h'}$ describes the sum of all next horizon sample values available from agent's current state, affected by observation function and reward functions, and is defined as*

$$\alpha'_{h'} = \sum_{u \in U} \alpha'_{h'} + R(h, u_i) \cdot T(u'_i|u_i, u'_i) \cdot O(h, u_i)$$

with h' as the next horizon, and u_i as the value of ith sample.

Transition function $T(s'|s, a)$ is defined as the probability to reach next state s', given the current state s and action taken a. In our case of motion planning, each of the horizon continuous state is a learning parameter, which are subjects of decision making. Therefore, because each learning parameter have to be chosen, the transition function $T(u'_i|u_i, u'_i)$ is redefined as the transition probability from current sample to the sample in the next horizon, taken the action which is sample in the next horizon itself, is always equal to 1.

Definition 6. (VALUE FUNCTION) *Given the belief function $b(h, u_i, o)$, reward function $R(h, u_i)$ and backup function $\alpha_{h'}$, value function is defined as*

$$V(b, u_i) = \sum_{o \in O} \sum_{u \in U} b(h, u_i, o) \cdot R(h, u_i) + \gamma \cdot \alpha'_{h'}$$

with h' as the next horizon, and u_i as the value of ith sample.

4.3 Adaptive Point-Based Value Iteration

In this section, imitation learning algorithm Adaptive Point-Based Value Iteration (APBIt) will be explained. Basically, the idea is to treat a continuous state as discrete states without permanent discretization, yet decides which actions to take by performing point-based value iteration. With n as the number of samples, *Algorithm 1* shows the main routine of the algorithm.

Algorithm 1 Main Routine of APBIt

```
1: function MAIN(h, U_n, O_n)
Require: h_min, I_m, I_o^n, S_i, γ
2:     U ← uniformSampling(h_min)
3:     for all o ∈ O_n do
4:         while h ≠ 0 do
5:             PBViteration(h, U_n, o)
6:             h ← h − 1
7:         end while
8:     end for
9: end function
```

APBIt performs asynchronous value function computation for each horizon based on sampling resolution. In our current progress, uniform sampling was employed, which will be improved in future works. V_{hor} which represents the value of each horizon, is initially set to be -1 so that any value compared with V_{hor} will be set as maximum value V_{max}. Variable h assigned as the number of total horizon, which will be as prerequisite for making a loop for every available horizon from t, for a t-step policy tree. Computation of each value function for each sample is performed by subroutine *PBViteration* in *Algorithm 2*.

Based on available samples, which in our current case performs uniform sampling, *PBViteration* simply calculates α, for all samples, continued with value function V_t computation. In our case of motion planning, we specify that the # of actions are equals to the # of states based on the sampling resolution. The sampling resolution can be adjusted according to needs.

A subroutine *calcVal* in *Algorithm 3* performs α computation, where α is the expected reward value. *back* function in *Algorithm 4* describes the iteration of ordinary α-vector backup for all samples.

Algorithm 2 Point-Based Value Iteration

function PBVITERATION(h, U_n, o)
  for all u $\in U$ **do**
    $\alpha[u] \leftarrow calcVal(h, u, U)$
    $V[u] \leftarrow b(h, u, o) \times \alpha(u)$
  end for
  $V_{\max} \leftarrow \max_{u \in U} [V[u]]$
  $s' \leftarrow \arg\max_{a \in A} [V[u]]$
  if $V_{\max} > V_{hor}[h]$ **then**
    $state_{hor}[h] \leftarrow s'$
    $V_{hor}[h] \leftarrow V_{\max}$
  end if
  return $state_{hor}[h]$
end function

Algorithm 3 Value Calculation

function CALCVAL(h, u, U)
  $\alpha' \leftarrow back(h, U)$
  $\alpha[u] \leftarrow R_{hor}(h, u) + \gamma \times \alpha'$
  return $\alpha[u]$
end function

5. Simulation Results and Discussion

5.1 Simulation Setup and Procedure

We simulate our algorithm in a static environment using V-Rep, a robot simulator, where the environment is assumed stay still during learning process. A 7-DOF manipulator with BarrettHand™Model was used as the agent. Two identical glass, one filled with water, are set on a table with the manipulator in initial position.It is assumed that we already get the input from the vision sensor which will determine the imitation result. Learning parameters involved comprises the X, Y, and Z-axis, which are the manipulator 3D coordinates workspace. The manipulator is assumed to be capable of performing a proper grasping of the glass and knows both position of glass. What we want to achieve is a proper pouring position after the glass filled in with water is grasped.

During the initial cycle, glass position will be a reference for pour position of the manipulator. Based on the glass position, our imitation learning algorithm tries to determine the pouring position as demonstrated by humans, which rewards functions are determined by the informations captured by the vision systems. A three-step policy tree was specified for the learning parameters, which each horizon comprises the X, Y and Z-axis,

Algorithm 4 Backup Operation

function BACK(h, u, U)
  $\alpha' \leftarrow 0$
  for all u $\in U$ **do**
    $\alpha' \leftarrow \alpha' + (R_{hor}(h-1, u) \times T() \times Z(h-1, u))$
  end for
  return α'
end function

Table 1 Learning Input Parameters

		Input (mm)		
Sensor		x	y	z
Main	min	35	40	36
	max	44	44	42
Observer 1	min	33	41	33
	max	40	44	40
Observer 2	min	31	37	38
	max	47	41	45
Observer 3	min	32	40	39
	max	42	47	46
Range	min	0	0	0
	max	150	300	150

shown in Figure 3. New learning parameter can be added in the decision making process below the range of Z-axis. In our algorithm, we need only to specify the range of the learning parameters. By performing a uniformly distributed random sampling according to predefined number of samples, our algorithm perform point-based value iterations, according to the samples obtained.

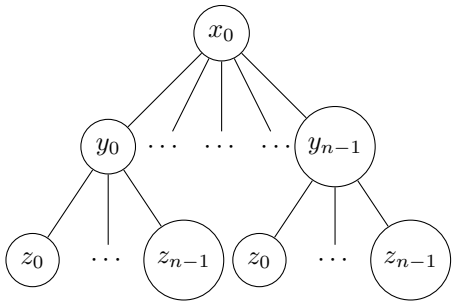
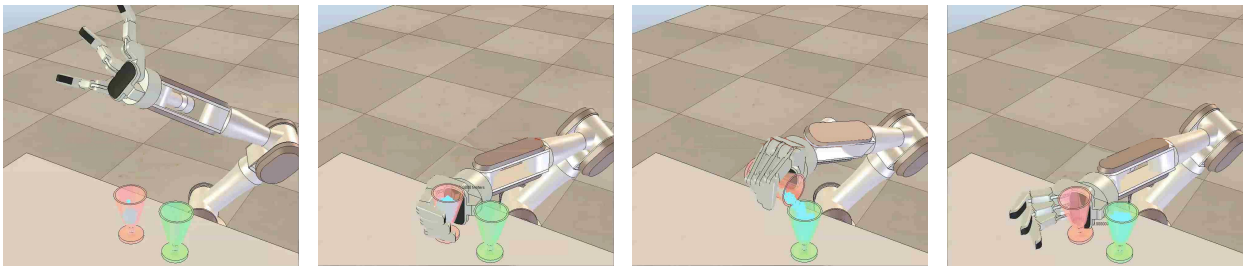


Fig. 3 Three-step policy tree of n-samples of learning parameters

5.2 Simulation Result

The input range from both main and observer sensors is to be assumed, shown in Table 1. We performed several separate simulations using a pseudorandom number generator (PRNG) with different amount of samples. Our algorithm was verified using 3D simulation, which result depicted in Figure 4 shows the performance using the converged output value, considering that the output of POMDP is the optimal policy tree π^*.

From Table 2, during 20 random samples, we can compare the output which shows the cartesian coordinate $(x, y, z) = (32, 32, 32)$ with the input from main and observer sensor. Overall, the output coordinate $(32, 32, 32)$ is not the optimal policy considering that the prediction of each sensors that the optimal policy lies somewhere between the (min, max) range (not necessarily in the middle). We notice that during 54 samples above, the output policy starts to converged. If we need to check the optimal policy for a certain input sets, all we have to do is just perform a fine discretization, which analogous to sample every single unit Since the smallest unit is $1mm$,



(a)Goes to intermediate position (b)Lift the glass and go to the position based on POMDP results (c)Delay to make sure every water particles have been poured (d)Release the glass and goes to intermediate position

Fig. 4 Pouring simulation result

Table 2 Simulation Result

	Output (mm)		
# of Samples	x	y	z
20	32	32	32
40	39	39	39
50	39	40	40
54	39	41	40
60	39	41	40
80	39	41	40
100	39	41	40
150	37	43	40
300	37	43	40

$Seed = 1.98$

we put 1 sample every $1mm$. When we tested with 150 samples or even 300 samples (which means for the range $(0, 150)$, every unit has 1 additional useless sample), the optimal policy for the assumed input sets are $(37, 43, 40)$. To analyze them, we defined the output of 300 samples as *the optimal policy* for the particular input sets. If we compare the output coordinates of 54 samples with the optimal policy, the x and y position only differs $2mm$, but the efficient samples needed to reach convergece is 54 samples. Therefore, we can say that *the suboptimal policy* is when the random samples is 54 for the particular input sets.

6. Conclusion

This paper presents new approach of imitation learning algorithm based on point-based value iteration which deals directly with continuous states POMDP using uniform sampling. Since our main objective is to perform a sequential decision making to achieve a sufficiently-good goal directed imitation learning, the simulation result shows that according to the sensor input (which is assumed in this case), even with uncertainty and a sufficient number of samples, we can perform sufficiently-good goal directed imitation learning. Various interesting things can be improved or added in this research, such as different grasping position even with the same target pouring position, adding more learning parameter such as pouring speed and angle, sampling optimization using MonteCarlo Method, or other numerical methods, considering feedbacks from human to improve the imitation performance.

References

[1] Cassandra, A.R. and Kaelbling, L P and Littman, M L, "Acting optimally in partially observable stochastic domains", *Proceedings of the National Conference on Artificial Intelligence*, pp. 1023–1023, 1995.

[2] Williams, J.D., "A case study of applying decision theory in the real world: POMDPs and spoken dialog systems", 2010.

[3] Hsu, D. and Lee, WS, "A point-based POMDP planner for target tracking", *International Conference on Robotics and Automation*, 2008.

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

[5] V-REP, 3D Robot Simulator, *http://www.coppeliarobotics.com*, may 2012.

[6] Bertsekas, Dimitri P, "Dynamic programming and optimal control", 1995.

[7] Littman, M L, "The witness algorithm: Solving partially observable Markov decision processes", PhD Thesis, Brown University, Providence, RI, 1994

[8] Roy, N and Gordon, G and Thrun, S, "Finding Approximate POMDP solutions Through Belief Compression", *Journal of Artificial Intelligence Research*, Vol. 23, 2005

[9] Thrun, S, "Monte Carlo POMDP", *Advances in Neural Information Processing Systems* Vol. 12, pp. 1064-1070, 2000

[10] Brooks, Alex M, "POMDPs for Planning in Continuous State Spaces", PhD Thesis, 2006

[11] Zhou, E. and Fu, M.C. and Marcus, S.I., "Solving continuous-state POMDPs via density projection", *IEEE Transaction on Automatic Control*, Vol. 55, NO. 5, MAY 2010

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