The 2016 AAAI Fall Symposium Series:
Artificial Intelligence for Human-Robot Interaction
Technical Report FS-16-01

Applying Adaptive Control in Modeling Human Motion Behaviors in Reinforcement Robotic Learning from Demonstrations

Huan Tan, Yang Zhao, and Balajee Kannan
GE Global Research, 1 Research Circle, Niskayuna, NY 12309, USA
huantan@ieee.org

Abstract

In this paper, we propose to use an adaptive control method as the basis of a reinforcement learning algorithm for robotic imitation learning. In the learning stage, robots use adaptive control method-based reinforcement learning algorithm to learn the parameters of dynamical systems. In the generation stage, robots use the learned dynamic system parameters and the pre-defined controller to drive the configuration states of the robot to move along desired state trajectories. One simulation experiment and one practical experiment on a robot are carried out to validate the effectiveness of our algorithm. The experimental results validate that the learning of the system parameters converges very fast and the learning results can improve the system performance of generating similar motion trajectories.

Introduction

Robotic Imitation Learning has been considered as a kind of power tools of transferring skills from humans to robots. From Uchiyama's experiment (Uchiyama 1978) in 1970s and Atkeson's trajectory learning experiment (Atkeson and McIntyre 1986) in 1980s, researchers proposed various methods for robots to learn behaviors (Argall et al. 2009), especially the motion trajectories, from humans. Figure 1 displays the basic idea of robotic imitation learning which is to model the knowledge or skills demonstrated by humans and to generate similar motion trajectories in similar but slightly different situations. These methods could be divided into two categories (Calinon, Guenter, and Billard 2007): one is to train robots to learn motion trajectories (Ijspeert, Nakanishi, and Schaal 2003), and the other is to train robots to learn behavior sequences (Dillmann, Kaiser, and Ude 1995). Either method involves system modeling or machine learning methods at the learning stage.

Some researchers believe that the offline learning methods could enable robots to generate similar motion trajectories after generalizing motion primitives (Ijspeert, Nakanishi, and Schaal 2003) (Peters and Schaal 2006). While some researchers consider it as a control problem for robots, which aims at driving configuration states of a robot to follow the states on a desired configuration trajectory (Arimoto 1990). The major difference between the two types of algorithms is how to modify the configuration of a robot. Conventionally, offline learning methods enable robots to have a precise mathematical model to describe the desired motion dynamics, which use a few parameters rather than a large number of data points and save storage space.

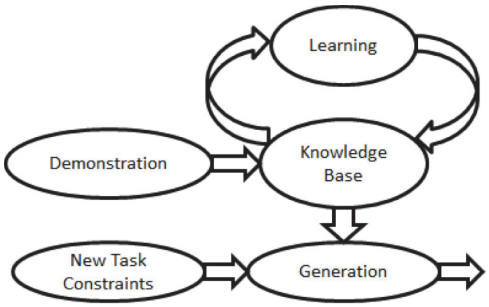

Figure 1 Imitation Learning Framework

In our research, we are interested in finding a control method for robots to track the desired motion trajectories. However, we do not know the internal dynamics of demonstrated motion trajectories. If we can teach a robot to learn the internal dynamics of a demonstrated motion trajectory, the motions will be more precisely modeled and generated. And we believe that the offline iteration learning could help robots learn the dynamics of the motion trajectories in advance.

In this paper, the motivation is to develop an offline algorithm to enable a robot to learn the internal dynamics feature of a demonstrated motion trajectory. Based on the learned results, the robot can use a control algorithm to generate control signals to drive the robot to follow the configuration states of a demonstrated motion trajectory.

The proposed method is largely based on the learning algorithm we choose. Current learning algorithms are normally categorized into three types (Bishop 2006): supervised learning, unsupervised learning, and reinforcement learning. Reinforcement Learning (RL) is widely applied in robotic imitation learning (Peters and Schaal 2008). Using RL methods, robots obtain rewards at each timing step or at the end of iterations and use such rewards to update the decision making policies. In imitation learning, the three types of algorithms are widely used in various domains. In this paper, we proposed to apply an adaptive control method in reinforcement learning for robotic imitation learning. Adaptive control (Astrom and Wittenmark 1994) (Li 1990) is a popular control method which is applied in dynamic systems with varying parameters. Upon the common feature in both adaptive control and the general robotic imitation learning, we believe that this control method is suitable for robots to improve policies through learning.

The rest of this paper is organized as follows: Section II reviews current research on robotic imitation learning; Section III derives a reinforcement learning algorithm based on adaptive control method; Section IV uses some experimental examples to explain how to apply this method in robotic imitation learning; Section V discusses the experimental results proposes the future work; and Section VI concludes this paper.

Related Work

A well accepted imitation learning framework can be described as: what to imitate, how to imitate, who to imitate, and when to imitate (Billard et al. 2007) (Calinon, Guenter, and Billard 2007). There are lots of replicating methods of generating motion trajectories the same as demonstrated motion trajectories. Readers can refer to this article (Argall et al. 2009) for more information. Adapting generation methods are of more interest in robotics community since it provides robots more flexible capability in various application domains.

Ijspeert proposed Dynamic Movement Primitives (DMP) (Ijspeert, Nakanishi, and Schaal 2003), which is considered as a tool for attractor landscape learning. A one-dimensional DMP formulation could be represented as:

$$\dot{z} = \alpha(\beta(g - y) - z) + f \quad (1)$$

$$z = \dot{y} \quad (2)$$

where g is the goal state, y is the position on the generated trajectory, z is the velocity, and f is a non-linear model. In original DMP, f is a regression model, which uses the weighted sum of basis functions to model the non-linear part of the motions. In our understanding, f is a non-linear regression model which modulates the system response.

Calinon and Billard proposed using Lagrangian methods (Calinon, Guenter, and Billard 2007) to minimize the distance between a demonstrated motion trajectory and generated motion trajectories in both the original data space and the dimension-reduced data space, called latent space. Adaptive methods used by Billard are also extended to Stable Estimator of Dynamical Systems (SEDS) (Neumann and Steil 2015), τ-SEDS (Khansari-Zadeh and Billard 2011), and Locally Modulated Dynamical systems (LMDS) (Kronander, Khansari, and Billard 2015).

In order to enable robot to learn policy parameters, e.g., the weights for f in equation (1) is learned through trials. Reinforcement learning is widely utilized by robotics community. Peters applied Reinforcement Leaning to teach robots to learn policy parameters or motion primitives though an iterative learning process (Peters and Schaal 2006). Thedorou proposed using optimal control for reinforcement learning applications (Theodorou, Buchli, and Schaal 2010). The charm of applying Reinforcement Learning is to provide a method for robots to find parameterized policies through trials. Other similar methods could be found in Ude's method (Ude 2010) of using Gaussian Process to update policy parameters, Billard's method (Billard 2007) of using Gaussian Mixture Model (GMM) to generalize demonstrations, etc.

Methodology

Background of Adaptive Control

Mechanical systems are generally described by nonlinear equations where Lagrangian and Hamiltonian mechanics are used to represent the behavior of mechanical system. A system with n Degrees-Of-Freedom (DOFs) can be locally represented by n configuration states and satisfies the following equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \quad (3)$$

where L is Lagrangian function denoting the total energy of the system:

$$L(q, \dot{q}, u) = T(q, \dot{q}) - V(q) + \sum_{i=1}^{n} q_i u_i \quad (4)$$

$$u_i = \tau_i \quad (5)$$

$T(q, \dot{q})$ is the total kinetic energy, $V(q)$ is the potential energy, and τ_i is the input variable.

By defining:

$$L(q, \dot{q}, u) = \dot{q}^T M(q)\dot{q}/2 - V(q) + q^T u \quad (6)$$

and after some mathematical derivations using equation (3), we could obtain:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (7)$$

where $M(q)$ is the inner mas matrix, $C(q, \dot{q})$ is related to Coriolis and centrifugal matrix, and $g(q)$ is the gravitational torque/force.

$$C(q, \dot{q}) = \dot{M}(q) - \frac{1}{2}\frac{\partial \dot{q}^T M(q)}{\partial q} \quad (8)$$

$$g(q) = \frac{\partial V(q)}{\partial q} \quad (9)$$

In some situations, we want to control this non-linear dynamical system to drive the state q to move along a desired trajectory in a configuration space or other relevant spaces. Using the known non-linear dynamics equation (7), we can design many types of controllers to generate desired input τ to control the plant.

A typical usage of adaptive control in this situation is to estimate the parameters in equation (7) and design a corresponding controller.

A required trajectory is represented as $(\ddot{q}_d, \dot{q}_d, q_d)$, where $q_d \in \Re^n$ is the positions on a desired trajectory, and $\dot{q}_d$ and $\ddot{q}_d$ are velocities and accelerations.

A proportional-differential (PD) controller is designed as:

$$\tau = \widehat{M}(q)\left[\ddot{q}_d - K_d(\dot{q} - \dot{q}_d) - K_p(q - q_d)\right] + \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q) \quad (10)$$

where $\widehat{M}(q)$, $\hat{C}(q, \dot{q})$, and $\hat{g}(q)$ are the estimates of M(q), $C(q, \dot{q})$, and $g(q)$.

If the estimates are exactly the same as actual parameter matrices, the closed loop dynamics is :

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \quad (11)$$

By selecting K_p and K_d suitably, the error is reduced to zero as time goes to infinity.

If the estimation is not perfect, the system dynamics is:

$$\widehat{M}(q)\left(\ddot{e} + K_d\dot{e} + K_p e\right) = -\widetilde{M}(q)\ddot{q} - \tilde{C}(q, \dot{q})\dot{q} - \tilde{g}(q) \quad (12)$$

where,

$$\widetilde{M}(q) = M(q) - \widehat{M}(q) \quad (13)$$

$$\tilde{C}(q, \dot{q}) = C(q, \dot{q}) - \hat{C}(q, \dot{q}) \quad (14)$$

$$\tilde{g}(q) = g(q) - \hat{g}(q) \quad (15)$$

The left hand side of equation (7) could be also described as:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Y(q, \dot{q}, \ddot{q})p \quad (16)$$

where $Y(q, \dot{q}, \ddot{q})$ is a regressor matrix and p is an unknown parameter vector.

Incorporating equation (7) (10) (13) (14) (15) (16), equation (11) could be written as:

$$\ddot{e} + K_d\dot{e} + K_p e = \widehat{M}(q)^{-1}Y(q, \dot{q}, \ddot{q})\tilde{p} \quad (17)$$

Define

$$\dot{x} = Ax - B\widehat{M}(q)^{-1}Y(q, \dot{q}, \ddot{q})\tilde{p} \quad (18)$$

where $x = [e^T, \dot{e}^T]^T$, $A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}$, and $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$.

In order to make the error converge to zero, we choose a Lyapunov function:

$$V(x, \tilde{p}) = x^T Px/2 + \tilde{p}^T Q\tilde{p}/2 \quad (19)$$

where Q is a positive definite matrix, and P is a solution to $A^T P + PA = -I$. By differentiating equation (19), we obtain:

$$\dot{V}(x, \tilde{p}) = -x^T Qx/2 - \tilde{p}^T\left[\left(\widehat{M}(q)^{-1}Y(q, \dot{q}, \ddot{q})\right)^T B^T Px + Q\dot{\tilde{p}}\right] \quad (20)$$

If the update law is:

$$\dot{\tilde{p}} = -Q^{-1}\left(\widehat{M}(q)^{-1}Y(q, \dot{q}, \ddot{q})\right)^T B^T Px \quad (21)$$

$$\dot{V}(x, \tilde{p}) = -x^T Qx/2 \leq 0 \quad (22)$$

the system error converges to zero.

This method is widely used in non-linear control of robotic manipulators.

Application in Reinforcement Learning

In robotic imitation learning, a robot should generate motion trajectories which are similar to demonstrated trajectories and the error mentioned above is not desired in the generation stage.

Some researchers try to train robots to modify policy parameters in a regression model to fit a desired trajectory. Then at the generation stage, give a goal state in a task, robots can generate similar trajectories to achieve the task goal. At the learning stage, the evaluation of modeling is over the whole desired trajectory, but not at each timing step. Therefore, the policy updating at the learning state should be considered as a result of considering the accumulated information over the whole desired trajectory.

We use a general dynamic system as shown in equation (7) and rewrite here:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (23)$$

The input τ is described as a regression function:

$$\tau = w_t p \quad (24)$$

where w_t is a regressor matrix, and p_t is a parameter vector.

During each trial, p is fixed. Only after each trial, the updates are integrated and this vector can be updated. In adaptive control algorithms, the parameters are updated in the control process which may cause the timing lag error. In our method, the parameters are fixed to accumulate the error over the whole trajectory for robots to learn. This could help robots to understand the whole demonstration avoiding the timing lag error.

We define a cost function as an integration of the rewards obtained over the generated trajectory.

$$R(\tau) = \phi(x_N) + \int_{t_i}^{t_N} r_t dt \quad (25)$$

where x_N is the state at the ending point of the trajectory, $r(t)$ is the reward, t_i is the starting time, and t_N is the ending time. τ represents the trajectory starting from t_i.

Simply, we can define

$$r_t = z_t + \tilde{p}_t{}^T Q\tilde{p}_t/2 \quad (26)$$

where z_t is an arbitrary reward function. Normally, it is the weighted square of the error between the desired trajectory (the demonstrated trajectory) and the generated trajectory.

$$z_t = x_t{}^T Px_t/2 \quad (27)$$

$$\phi(x_N) = z_{t_N +}\tilde{p}_{t_N}{}^T Q\tilde{p}_{t_N}/2 \quad (28)$$

According to the updated law in equation (21):

$$\dot{\hat{p}} = -Q^{-1}\left(\widehat{M}(q)^{-1}w_t\right)^T B^T Px \quad (29)$$

where $x = [e^T, \dot{e}^T]^T$, $A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}$, and $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$.

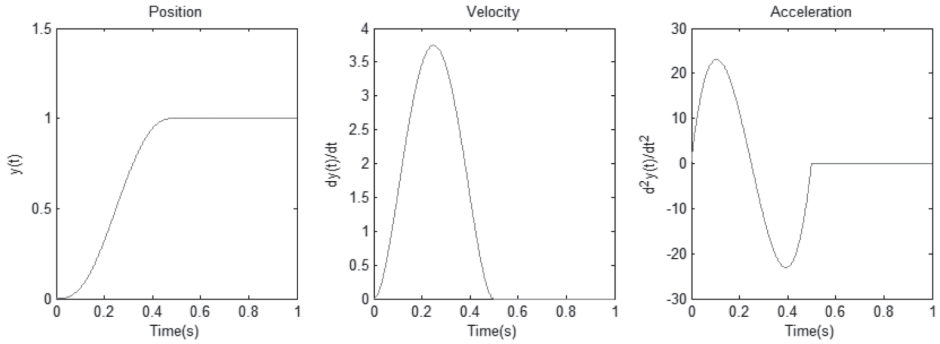

Figure 2 Demonstrated Motion Trajectory

The overall updates can be computed as the integration of update at each timing step over the generated trajectory:

$$\delta p = \int_{t_i}^{t_N} \left(-Q^{-1}\left(\widehat{M}(q)^{-1}w_t\right)^T B^T P x\right) dt \quad (30)$$

Then, we get

$$p_{new} = \left(p_{old} + \int_{t_i}^{t_N} \left(-Q^{-1}\left(\widehat{M}(q)^{-1}w_t\right)^T B^T P x\right) dt\right) \quad (31)$$

Application in Parameterized Motor Primitives Learning

In order to adaptively generate motion trajectories in similar but slightly different situations, we pick DMP as the basis for describing internal dynamics of motions. As explained earlier, f modulates the system response. In our understanding, f can also be considered as the controller input which modulates or drive the system response.

In original DMP, α and β are predefined arbitrarily to ensure the over damping performance of the motion generation system. In this paper, we applied Adaptive Control to enable robots to learn the two parameters iteratively.

Rewrite equation (1) and equation (2), and incorporate equation (16):

$$\ddot{y} + \alpha\dot{y} + \alpha\beta y - \alpha\beta g = Y(y, \dot{y}, \ddot{y})\theta \quad (32)$$

By replacing $y - g$ with q, we can obtain that:

$$\ddot{q} + \alpha\dot{q} + \alpha\beta q = Y(q, \dot{q}, \ddot{q})\theta \quad (33)$$

In order to apply adaptive control-based reinforcement learning for DMP, we construct the regressor matrix as:

$$Y(q, \dot{q}, \ddot{q}) = [\ddot{q}\ \dot{q}\ q] \quad (34)$$

The parameter vector is:

$$\theta = [1\ \alpha\ \alpha\beta]^T \quad (35)$$

In this equation,

$$M(q) = 1 \quad (36)$$

$$C(q, \dot{q}) = \alpha \quad (37)$$

$$g(q) = \alpha\beta q \quad (38)$$

The controller is designed as:

$$f = \tau = \left[\ddot{q}_d - K_d(\dot{q} - \dot{q}_d) - K_p(q - q_d)\right] + \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q) \quad (39)$$

We can obtain the system dynamics equation

$$\ddot{e} + K_d\dot{e} + K_p e = Y(q, \dot{q}, \ddot{q})\tilde{\theta} \quad (40)$$

Define

$$\dot{x} = Ax - BY(q, \dot{q}, \ddot{q})\tilde{\theta} \quad (41)$$

where $x = [e^T, \dot{e}^T]^T$, $A = \begin{bmatrix} 0 & 1 \\ -K_p & -K_d \end{bmatrix}$, and $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and choose the Lyapunov Function

$$V(x, \tilde{p}) = x^T P x/2 + \tilde{\theta}^T Q \tilde{\theta}/2 \quad (42)$$

where Q is a positive definite matrix, and P is a solution to

$$A^T P + PA = -I \quad (43)$$

The update law is:

$$\dot{\hat{\theta}} = -Q^{-1}Y^T B^T P x \quad (44)$$

Using the principle of reinforcement learning, at the end of each trial, θ_{new} could be computed as

$$\theta_{new} = \left(\theta_{old} + \int_{t_i}^{t_N}(-Q^{-1}Y^T B^T P x)dt\right) \quad (45)$$

Through trials, θ can be updated iteratively. Then, given a new task goal, a motion trajectory could be generated using the controller defined by equation (1), (2), and (39).

Experimental Results

In order to validate our proposed algorithm, we designed two experiments in simulation and one experiment on a robot. We want to validate that the robot can learn the dynamics, i.e., the model parameters, and generate similar motion trajectories.

The demonstrated motion trajectory used in Experiment 1 and 2 is displayed in Figure 2. The spring-damper dynamics of equation (33) is determined by α and $\alpha\beta$. If we rewrite left side of the equation (33) in S-domain, we can use linear system analysis method to find the dynamics. A standard 2-order linear system can be described as

$$G(s) = \frac{{\omega_n}^2}{s^2 + 2\xi\omega_n + {\omega_n}^2} \quad (46)$$

The denominator of the $G(s)$ determines the dynamics of a system. An important feature is that ξ determine the damping type of the system.

In order to avoid the damage caused by damping and to ensure that the response is fast enough to follow the desired motion, normally, we want to construct a critical-damping system. The weight matrix in equation (42) is defined as a positive identity matrix $\boldsymbol{I}$. The proportional and differential parameters are arbitrarily chosen as:

$$K_p = 400 \quad (47)$$

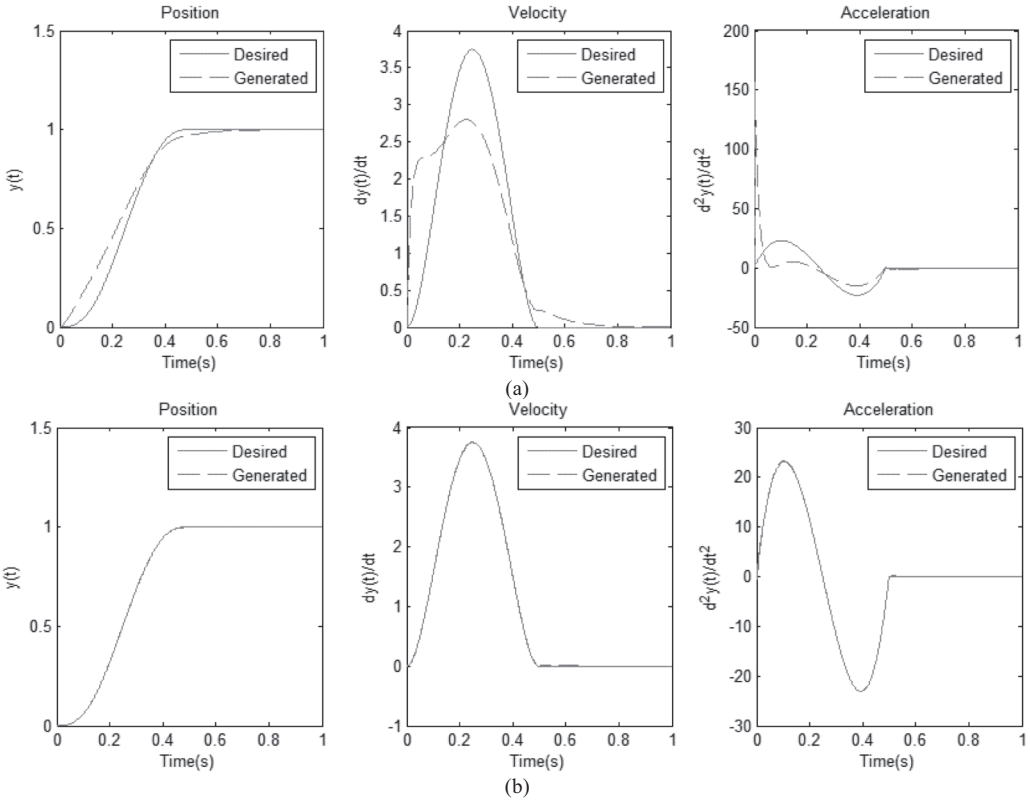

Figure 3 Generated Motion Trajectory in Experiment 1

$$K_d = 40 \tag{48}$$

The transition matrix is:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -400 & -40 \end{bmatrix} \tag{49}$$

After solving equation (43), we get

$$\boldsymbol{P} = \begin{bmatrix} \frac{405}{80} & \frac{1}{800} \\ \frac{1}{800} & \frac{401}{32000} \end{bmatrix} \tag{50}$$

Experiment 1 and 2 use the same parameter matrices in the learning processes as explained above.

Experiment 1 (Simulation)

The dynamics parameters are chosen as follows:

$$\alpha = 25 \tag{51}$$

$$\beta = 6.25 \tag{52}$$

Then the damping ratio is $\xi = 1$.

The initial estimation of the parameters are:

$$\alpha_0 = 0 \tag{53}$$

$$\beta_0 = 0 \tag{54}$$

Without learning and using the initial parameters, the generated motion trajectory using equation (1) is displayed in Figure 3-(a). The blue solid lines are the demonstrated motion trajectory and are also considered as desired motion trajectory, and the red dashed lines are generated motion trajectory. From Figure 3-(a), the generated motion trajectory has large deviation from the desired motion trajectory, due to the fact that the parameters α and β in the controller are not the same as the parameters used in equation (1). After learning, the generated motion trajectory is shown in Figure 3-(b), which is overlapped with the demonstrated motion trajectory. This means that the generated motion trajectory is similar to the demonstrated motion trajectory after learning. And the learned parameters are:

$$\alpha_N = 24.9443 \tag{55}$$

$$\beta_N = 6.3074 \tag{56}$$

The estimated parameters converge to the actual parameters after 200 iterations. Theoretically, when $t \to \infty$, the estimated parameters converge to the actual parameters.

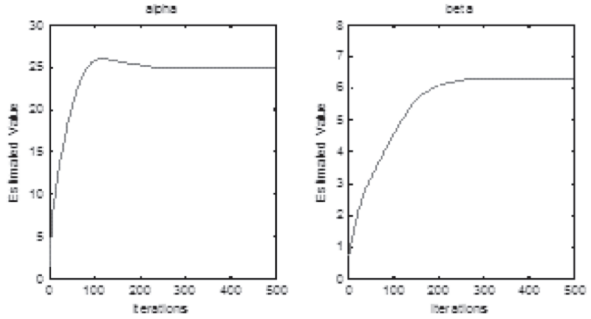

Figure 4 Estimated Parameters in Experiment 1

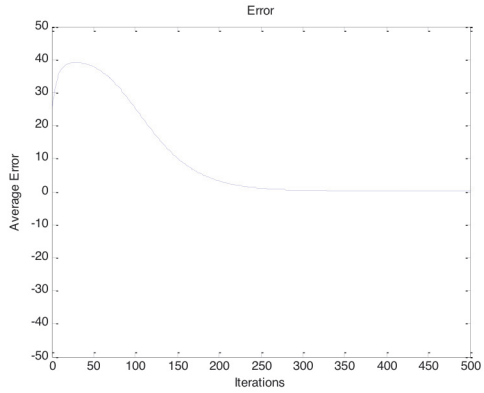

Figure 5 Error Values of Experiment 1

The error value is defined as the distance between the generated motion trajectory and the demonstrated motion trajectory. Mathematically, it is computed as:

$$e = \int_{t_0}^{t_N} abs\big(y^d(t) - y(t)\big)dt \qquad (57)$$

Figure 5 displays the error values in iterations. As shown in Figure 5, the error value converges to zero after 200 iterations which reflect the robustness and rapid convergence of our algorithm. Figure 4 and Figure 5 can be also considered as the learning curve in our experiments.

After the parameters have been learned, they can be directly used in DMP to generate similar motions. Even with different initial positions and target positions, the dynamics of motion trajectories will be the same.

Experiment 2 (Robotic Conducting)

We used this method to teach a robot to learn Conducting motion trajectory. Figure 6 displays a typical conducting trajectory and the learning results. From Figure 6, we can see that the robot successfully learned the Conducting behavior. The learning curve is similar to Figure 5.

Discussion and Future Work

From the experimental results, we can conclude that our algorithm is robust and the convergence speed of learning is fast. After 200 iterations, the learning of the parameters converges to the actual value of the parameters in the dynamic system we use. Meanwhile, the error values drops to zero at the same speed as the learning.

However, the learning at the beginning is not the same as conventional learning algorithms. Especially, the learning curve for alpha has an overshoot at the beginning around iteration 100, which correspondingly causes the overshoot of error values on the learning curve. The reason for the overshoot is that the learning of the two parameters is separated since the matrix Q chosen in this paper is diagonal. The correlation between the two parameters is arbitrarily removed. Our future work is to find a method to determine the correlation values, i.e. the off-diagonal numbers in the matrix.

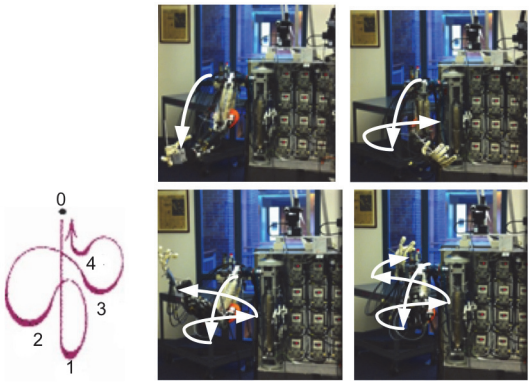

Figure 6 Learning Result of Experiment 2

As mentioned earlier, this method firstly determines the type of controller and then uses the iteratively estimated parameters in the dynamical system to adaptively update the controller, which is the basic idea of adaptive control. Through iterative learning, the learned model can enable robot to generate motion trajectories similar to demonstrations. Compared to other reinforcement learning algorithms applied in robotic imitation learning, this algorithm can largely save the time of learning. As shown in the experimental results, after 200 iterations, the estimated parameters converge to the actual values that we use in our dynamical systems. This method provides an extra solution to current robotic imitation learning research.

Conclusion

This paper proposes applying adaptive control method in reinforcement learning for robotic imitation learning. Experimental results given in this paper validate that the proposed algorithm is robust and enable a robot to learn motions demonstrated by human teachers quickly and reliably. The proposed algorithm contributes to current robotic imitation learning and reinforcement learning research community and provides an extra solution for robots to learn motion trajectories demonstrated by human teachers.

References

Argall, B. Chernova, S, Veloso, M, and Browning, B. 2009. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems* 57: 469-483.

Arimoto, S. 1990. Robustness of learning control for robot manipulators. In *Proceedings of 1990 IEEE International Conference on Robotics and Automation*: 1528-1533. Cincinnati , Ohio, USA. IEEE Press.

Astrom, K. J. and Wittenmark, B. 1994. *Adaptive control*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Atkeson, C. and McIntyre, J. 1986. Robot Trajectory Learning Through Practice. In *Proceedings of 1986 IEEE Conference on*

Robotics and Automation, 1737-1742, San Francisco, California, USA. IEEE Press.

Billard, A. Calinon, S. Dillmann, R. and Schaal S. 2007. Robot programming by demonstration, in *Handbook of robotics*, B. Siciliano and O. Khatib, Eds., ed. Springer, New York, NY, USA.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Calinon, S., Guenter, F. and Billard, A. 2007. On learning, rep resenting, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37: 286-298.

Dillmann, R. Kaiser, M. and Ude, A. 1995. Acquisition of elementary robot skills from human demonstration. In *Proceedings of 1995 International Symposium on Intelligent Robotic System*: 185-192. Pittsburgh, Pennsylvania, USA. IEEE Press.

Ijspeert, A. Nakanishi, J. and Schaal, S. 2003. Learning attractor landscapes for learning motor primitives. *Advances in Neural Information Processing Systems* 15: 1523-1530.

Khansari-Zadeh, S. and Billard, A. 2011. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics* 27: 943–957.

Kronander, K. Khansari, M. and Billard. 2015. A. Incremental motion learning with locally modulated dynamical systems. *Robotics and Autonomous Systems* 70: 52–62.

Li, W. 1990. Adaptive control of robot motion, Ph.D Dissertation, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Neumann, K. and Steil, J.. 2015. Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems* 70: 1–15.

Peters, J. and Schaal, S. 2006. Reinforcement learning for parameterized motor primitives. In *Proceedings of 2006 International Joint Conference on Neural Networks*: 73-80. Vancouver, BC, Canada. IEEE Press.

Peters, J. and Schaal, S.. 2008. Learning to control in operational space. *The International Journal of Robotics Rese arch* 27: 197.

Theodorou, E. Buchli, J. and Schaal S. 2010. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research* 11: 3137-3181.

Uchiyama, M. 1978. Formation of High Speed Motion Pattern of Mechanical Arm by Trial. *Transactions of Society of Instrument and Control Engineers* 19: 706-712.

Ude, A. Gams, A. Asfour, T. and Morimoto J. 2010. Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives. *IEEE Transactions on Robotics* 26: 800-815.