Third International Conference on
Advances in Control and Optimization of Dynamical Systems
March 13-15, 2014. Kanpur, India

Robot learns from human teacher through modified kinesthetic teaching

D. P. Ghoshal, N. Das, S. Dutta, L. Behera

Department of Electrical Engineering, Indian Institute of Kanpur, India.

Email: {dghoshal, niladri, samratd, lbehera}@iitk.ac.in

Abstract: Teaching new motor tasks to robots through physical interactions is an important goal for both robotics and machine learning. Most monolithic machine learning approaches fail to scale when going beyond basic skills. In this paper we present a simple framework for teaching the robot (to play tennis) through direct physical interaction with a human teacher (i.e. Kinesthetic Teaching). Current popular established method of kinesthetic teaching generally uses a two-stage approach: First, a library of motor primitives is generated through direct physical manipulation of the robot. In second stage, a reinforced learning ("reward" stage) is implemented to dynamically adjust the policy of choosing from motor primitive library. In this paper, we show that by proper modification of the first stage of Kinesthetic Teaching and incorporating the domain experience of the human teacher, we can remove the necessity of the second stage. This approach has multiple advantages: (i) We can make the whole training process much simpler. This would go a long way in making our training algorithm scalable (for much increased number of basic moves, etc). (ii) One potential problem with the "reward" learning phase is that there may be subjective difference of what is a "good" shot from the kinesthetic teaching and from the bystander viewpoint (later in "reward" stage). Even a little difference in this regard will result in a confusing feedback ("reward"), and hence it would be difficult to correctly figure out which library training samples should be reassigned what weight values. Our approach eliminates this problem altogether.

Keywords: robot learning, learning by demonstration, kinesthetic teaching, motor skill learning, generalizing movements.

1. INTRODUCTION

The basic difference between humans and robots in learning complex motor tasks is that humans can generalize the movement and adapt to changing context as and when needed. Robots, on the other hand, rely heavily on well-modelled environment and rigid parameters. Enabling a robot to develop new motor tasks (as required by the change in context) has been a challenge for a quite a time (Schaal et al., 2002).

Humans learn complex motor skills by decomposing them into smaller subtasks. This concept can be used for reducing the complexity involved in teaching robots new motor skills. The subtasks can often be solved by use of a relatively small number of Movement Primitives (Flash and Hogan, 1985; Giszter et al., 2000; Billard et al., 2008). Movement primitives are stream of motor commands implemented to fulfil a given motor task.

With the increase in number of dimensions and time steps, the number of scenarios to be explored increases exponentially (Schaal, 1999) – hence the need of efficient learning techniques of movement primitives. Numerous studies indicate that efficient acquisition of single movement primitives can be done through Learning by demonstrations (Guenter et al., 2007; Peters and Schaal, 2008b; Kober et al., 2008; Peters and Schaal, 2008a; Bitzer et al., 2010).

Most schemes for robot imitation learning use either a motion capture system (Kober et al., 2008), or physical demonstrations of a movement to the robot by a human teacher – Kinesthetic Teaching (Guenter et al., 2007; Peters and Schaal, 2008b,a; Bitzer et al., 2010; Mulling et al., 2013).

Current popular established method (Mulling et al., 2013) of kinesthetic teaching generally uses a two-stage approach: First, a library of movement primitives is generated through direct physical manipulation of the robot. In second stage, a reinforced learning ("reward" stage) is implemented to dynamically adjust the policy of choosing from movement primitive library. In this paper, we show that by proper modification of the first stage of Kinesthetic Teaching and incorporating the domain experience of the human teacher, we can remove the necessity of the second stage.

In the rest of the paper, we proceed as follows. In section 2, we describe the experimental setup. Section 3 details the process of ball detection in image space. Section 4 describes the method used for predicting the ball flight trajectory. In section 5, we explain our approach of teaching the robot the arm motions. In section 6, we present our results. Finally, conclusion and future work is discussed in section 7.

2. EXPERIMENTAL SETUP

This section describes the experimental setup. The setup includes a mobile robot platform (Robotnik Guardian) which has an attached robotic arm (Barrett WAM) with seven

Degree-of-freedom capability. Microsoft Kinect is used as the visual sensor device. A lightweight end-effecter is attached to the WAM arm with which the ball would be hit. The ball is served to the forehand of the WAM arm by a human. The total setup area is approximately 25 m^2. The whole setup operates under the ROS (Robot Operating System) framework.

3. BALL DETECTION IN IMAGE-SPACE

This section describes the ball-tracking module. OpenCV and OpenNI have been used under ROS framework to implement this module.

OpenCV is a standard open-source computer vision software. It contains a collection of image processing functions aimed at real-time operation. It also provides some generic interface to various 2-D image sensors.

OpenNI is a software framework containing libraries which acts as a middleware between the 3-D sensor (KINECT) and its applications. One can use these libraries as a standalone package for working with the sensor or one can use the same libraries as a software package inside the ROS framework.

Robot Operating System (ROS) is a framework containing various software packages like sensor drivers (e.g. OpenNI), image processing packages (e.g. OpenCV, Point Cloud Library) and others. It offers a platform which integrates different software packages related to robot operation. The most important feature of it is the seamless message transfer between different software packages as well as between different nodes (robots having ROS installed).

In our work the OpenNI package inside ROS was used to communicate with the Kinect hardware. It published image data in form of messages to the entire network of ROS nodes. The images were obtained by subscribing to those messages and then OpenCV was used to process them.

During the flight-time of the ball, stream of images are captured through Kinect device. Kinect provides two types of image information: RGB (Red-Green-Blue) image and Depth image. Both type of images are used and processed to get the ball trajectory information. The steps are discussed below.

3.1 Transformation to HSV domain

The RGB images were transformed to HSV (Hue-Saturation-Value) domain through standard OpenCV functions. Figure 1 shows the image of the ball in RGB domain while Figure 2 shows that in HSV domain. Using only the Hue channel from the HSV domain makes it easier to identify the ball from the background image.

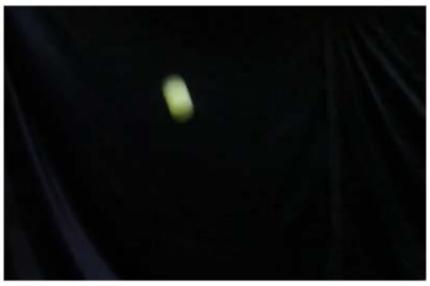

Figure 1. RGB image of the ball

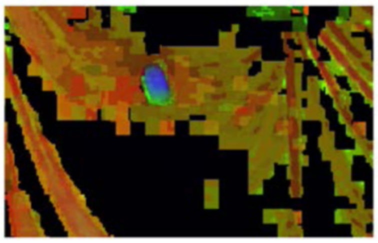

Figure 2. Image in HSV domain

3.2 Thresholding

Using only the Hue domain data, thresholding was done on the HSV image to convert it into binary image. The thresholding value was tuned according to the ball colour (green) and the background (black).

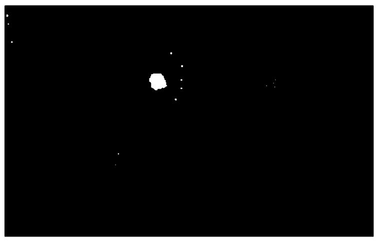

Figure 3. Image after binary thresholding

3.3 Noise reduction

The binary image obtained after thresholding contains noise in form of stray white dots on black background (shown in Figure 3). Successive erosion and dilation were applied over the noisy image to remove the stray white dots. For implementing these morphological transformations, two masking matrices of different sizes were used - One for erosion and one for dilation. These masks are available as standard OpenCV functions. The required sizes of these masks were selected heuristically. Figure 4 shows the image after noise reduction.

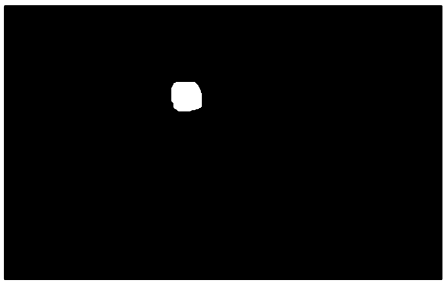

Figure 4. Image after noise reduction

3.4 Contour & centroid detection

The contour of the white patch (representing the ball) was detected and the centroid of that white patch was obtained through first-order moment calculation. This gives the co-ordinate (in 2D plane) of the ball at that instant. The + sign in Figure 5 shows the obtained centroid.

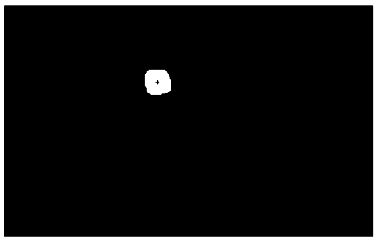

Figure 5. Ball detection & centroid calculation

3.5 Depth information

Depth images corresponding to the HSV images are obtained from the Kinect device. The slight offset in the two images (due to the difference of the RGB sensor and depth sensor positions) are automatically nullified through standard OpenNI functions. The depth value at the ball co-ordinate (already calculated) was obtained. Thus, all 3 co-ordinates of the ball at that particular instant are obtained along with the corresponding timestamp. Repeating the process for every snapshot of the video stream, the ball trajectory was tracked throughout its flight.

The overall steps for the ball detection are presented as a flowchart in Figure 6.

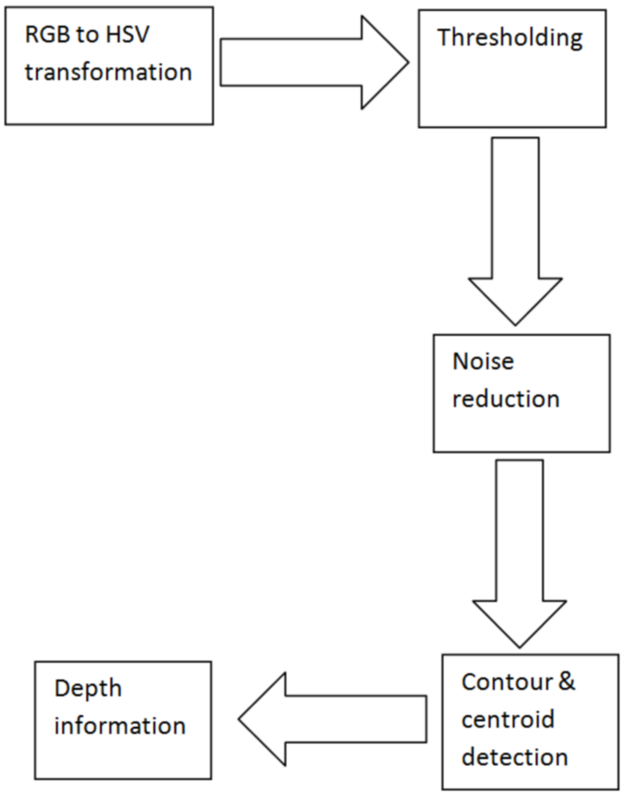

Figure 6. Flowchart for ball detection process

4. BALL TRAJECTORY PREDICTION

The set of complete trajectories obtained from tracking of numerous ball throws was used as sample data for estimating the parameters of a modified projectile motion equation. Once the equation parameters were tuned, then it was used for predicting the trajectory of a new ball-throw. Using different measurement points from the initial flight path of the ball, several curves are calculated, and then these curves are averaged to produce the final predicted trajectory of the ball. The coordinate frame used is shown in Figure 7.

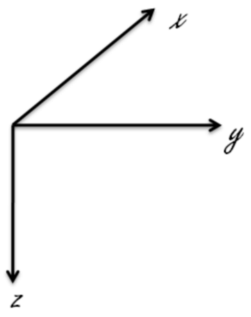

Figure 7. Coordinate frame used for trajectory prediction

The differential equation used is similar to the generalized 3D projectile motion but with an added term representing air-drag. The drag-force has been approximated to be varying proportional to the speed of the ball. Since the motions along x, y, z directions are decoupled, we can write three independent differential equations.

$m\ddot{x} = -k_1 v_x$

$m\ddot{y} = -k_2 v_y$

$m\ddot{z} = -k_3 v_z + mg$

Where,

m = mass of the ball

x, y, z = position of the ball along the three axes

v_x , v_y , v_z = speed of the ball along the three axes

g = acceleration due to gravity

k_1 , k_2 , k_3 = drag coefficients along the three axes

The solutions of these equations are used in the trajectory prediction.

$$x(t) = x(t_0) + \frac{k_1}{m} v_x(t_0) - \frac{k_1}{m} v_x(t_0) e^{\frac{k_1}{m}(t_0 - t)}$$

$$y(t) = y(t_0) + \frac{k_2}{m} v_y(t_0) - \frac{k_2}{m} v_y(t_0) e^{\frac{k_2}{m}(t_0 - t)}$$

$$z(t) = z(t_0) + \frac{k_3}{m} v_z(t_0) - \frac{k_3}{m} v_z(t_0) e^{\frac{k_3}{m}(t_0 - t)} + \frac{mg}{k_3}(t - t_0) + g\left(e^{\frac{k_3}{m}(t_0 - t)} - 1\right)$$

Where,

t_0 = initial time

5. TRAINING

5.1 Imitation Learning (IL) and Reinforcement Learning (RL)

One of the major disadvantages of Imitation Learning is that the robot's performance is limited by the human teacher's demonstration – the robot cannot improve upon them, and hence cannot improvise when faced with a changing context. On the other hand, Reinforcement Learning works through free exploration of state-action space, and thus allows the robot to discover new control policies. But using Reinforcement Learning alone often results in long converging times.

Complex motor tasks (such as playing tennis) are difficult to teach to a robot by using either of the two learning methods alone. A combination of IL and RL has been used in similar cases (Kober and Peters, 2011; Kormushev et al., 2010; Jetchev and Toussaint, 2013; Bentivagna et al., 2004; Mulling et al., 2013). In the latter case, an initial set of movement primitives are generated through IL, and then RL is used to develop a policy for appropriate selection from the primitive library.

But this approach makes the whole training process a two-stage affair, and hence effectively not scalable. Another potential problem with the Reinforcement learning phase is that there may be subjective difference of what is a "good" shot from the kinesthetic teaching and from the bystander viewpoint (later in RL stage). Even a little difference in this regard will result in a confusing feedback, and hence it would be difficult to correctly figure out which library training samples should be reassigned what weight values.

Thus we have developed a different approach wherein the expert knowledge of the human teacher is incorporated in the first phase (creation of movement primitive library) itself. By correctly assigning different priorities (by the human expert) to the movement primitives of the library, the second phase (Reinforcement learning) becomes redundant in most of the cases. This in turn makes the overall training process much simpler and thus much more scalable.

5.2 Creating the movement primitives library

The movement primitive library was created by recording successful hitting of the ball by the WAM arm guided by a human teacher (kinesthetic teaching). The ball trajectory was tracked and recorded and the corresponding "ideal" shot (as played by the human teacher by physically moving the WAM arm) was recorded as a movement primitive. The WAM arm movement was measured by logging actual WAM data of joint positions and joint velocities. Repeating this process for different throw-and-hits, the movement primitive library was generated.

5.3 Offline training of the gating network

The sample database of the movement primitive library was used to train Artificial Neural Network (ANN) to work as a gating network for generating new arm movements.

ANN is a computational model inspired from biological neural networks. ANN can be used to detect patterns from any particular category of data through proper training using sample data sets i.e. ANN has the ability to learn and generalize - this generalization capability of ANN is used in this study.

Here the ANN uses the standard sigmoid function as the activation function. The number of neurons and hidden layers are determined according to the number of movement primitives in the library.

5.4 Generation of the required striking motion

When a ball is thrown, its motion is tracked and the complete future trajectory is predicted through the prediction module. This trajectory is then passed on to the gating network, which then produces different "weights" corresponding to the different movement primitives of the library. These weight values, combined with the priority values (as set by the human teacher) are used to generate the ideal arm motion for that particular ball throw.

6. SIMULATION RESULTS

The ball-trajectory prediction module was first evaluated. Ball trajectory was predicted by using the data captured from the initial stage of the flight path, and then compared against the actual flight trajectory of that ball-throw.

Figure 8 shows one such comparison. The straight line represents the predicted trajectory, and the stream of circles represents the actual ball trajectory as recorded by Kinect.

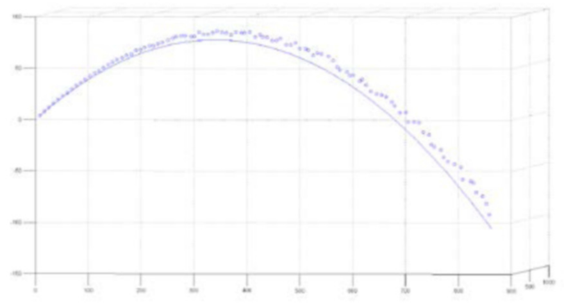

Figure 8. Predicted ball trajectory vs. Actual ball trajectory

For evaluation of the arm motion, test cases were recorded where the human teacher gave the demonstration of a successful hit by physically moving the arm for a particular ball-throw. Joint positions and joint velocities of the arm were collected. Then an "ideal" arm motion was generated (in terms of joint positions and joint velocities) from the movement primitive library using the learning policy. This motion was then compared to the actual successful demonstration motion. The comparison was done in joint position space, joint velocity space, and also in Cartesian space. For transformation from joint space to Cartesian space, Denavit and Hartenberg matrices were used.

Figures 9 through 15 show such a comparison case of joint position values for joints 1 through 7 respectively. The solid line represents the predicted data, and the dashed line represents the actual data.

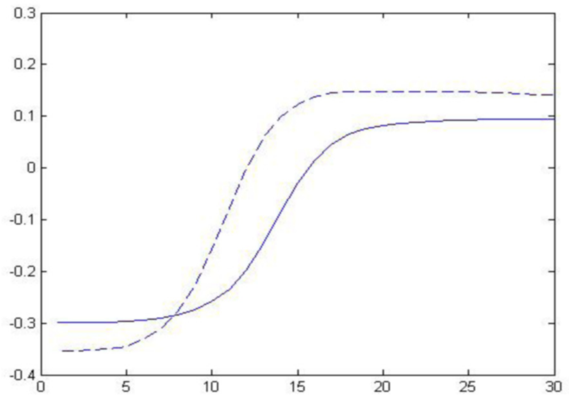

Figure 9. Variation of Joint 1 angle with time

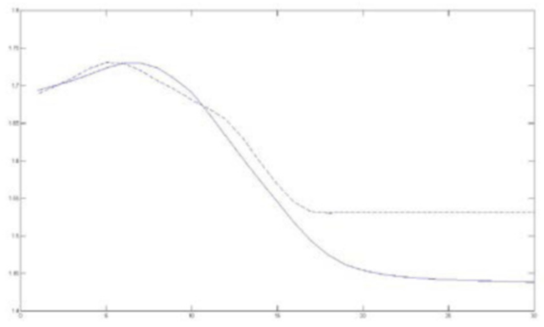

Figure 10. Variation of Joint 2 angle with time

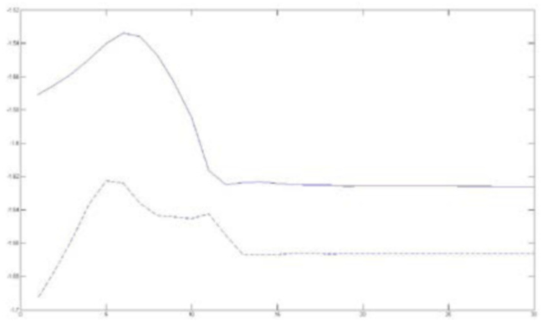

Figure 11. Variation of Joint 3 angle with time

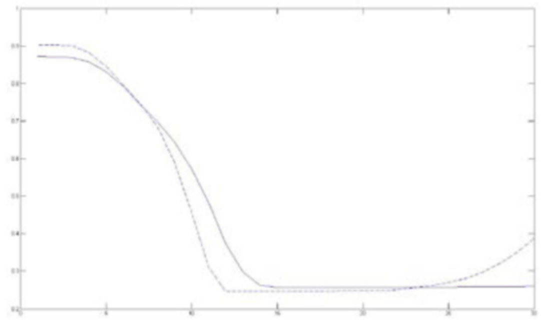

Figure 12. Variation of Joint 4 angle with time

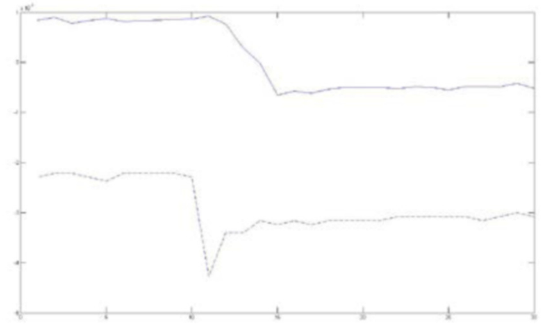

Figure 13. Variation of Joint 5 angle with time

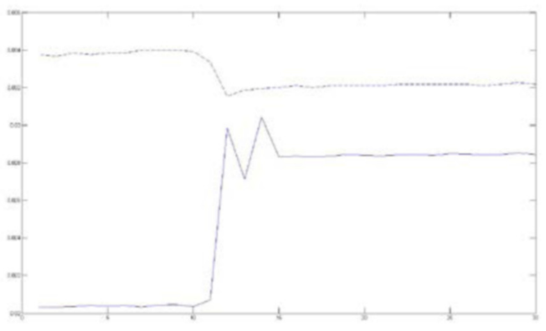

Figure 14. Variation of Joint 6 angle with time

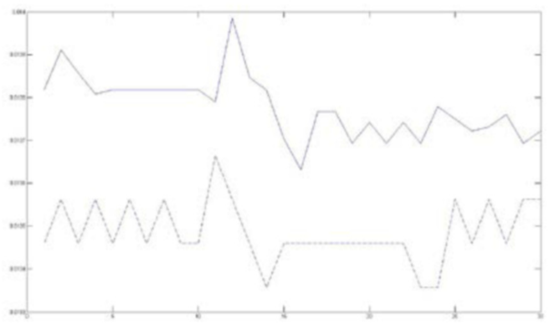

Figure 15. Variation of Joint 7 angle with time

Figures 16 through 22 show comparisons of joint angular velocity values for joints 1 through 7 respectively. The solid line represents the predicted data, and the dashed line represents the actual data.

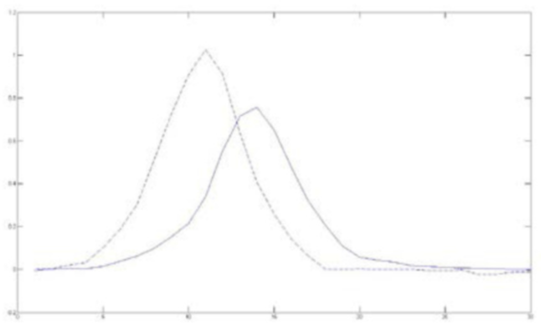

Figure 16. Variation of angular velocity of Joint 1 with time

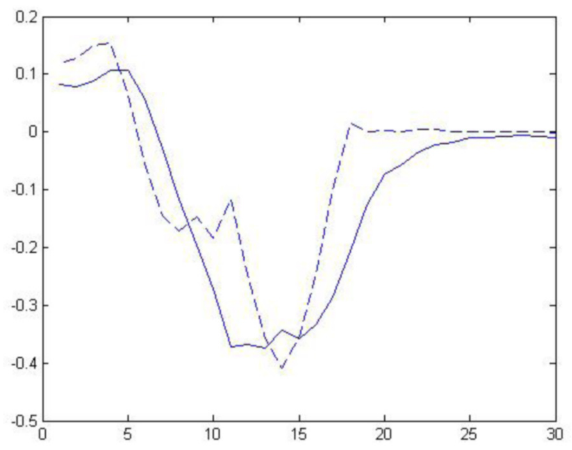

Figure 17. Variation of angular velocity of Joint 2 with time

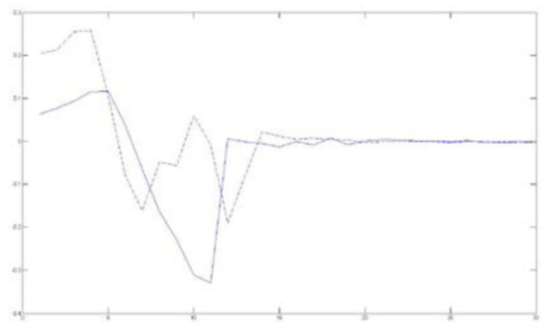

Figure 18. Variation of angular velocity of Joint 3 with time

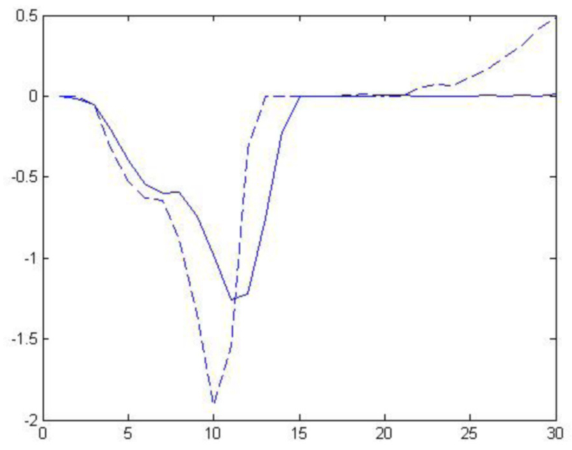

Figure 19. Variation of angular velocity of Joint 4 with time

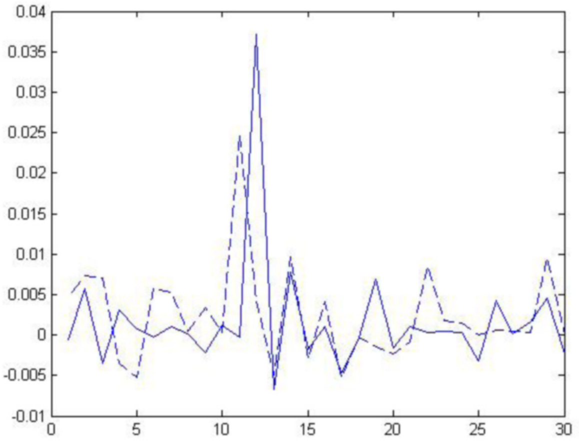

Figure 20. Variation of angular velocity of Joint 5 with time

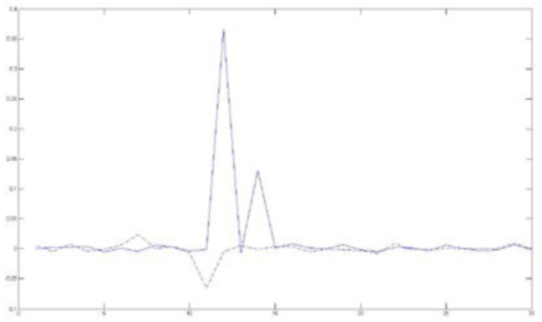

Figure 21. Variation of angular velocity of Joint 6 with time

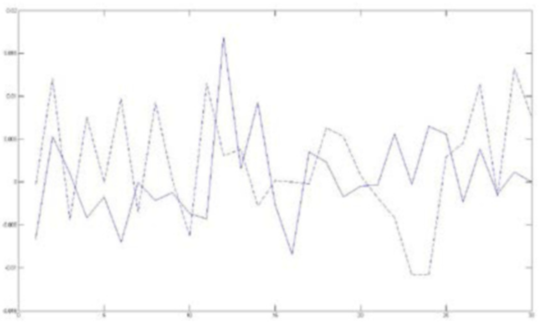

Figure 22. Variation of angular velocity of Joint 7 with time

Though some of the figures show some offset, it should be noted that differences are quite negligible in magnitude.

Figure 23 shows the comparison between the generated and demonstrated path of the arm end-effecter in Cartesian space for one test case. Blue line represents the demonstration path and red line represents the generated path.

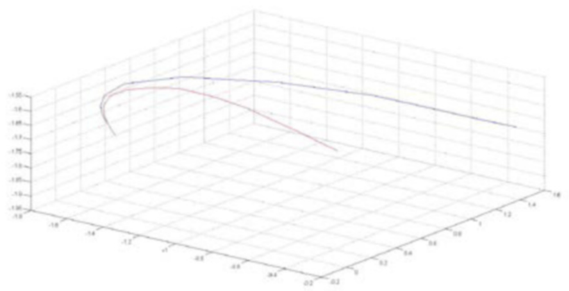

Figure 23. Comparison between generated & demonstrated path of robotic-arm end-effecter in Cartesian space

The offset at the beginning of the motion is due to the fact that the human teacher started the shot from a back-lift position while the generated path started from a rest position. The important point to be noted is that at the end of the movement, the generated path almost converges with the demonstration path, and hence can be taken as a "successful" hit.

7. CONCLUSION

The simulation results indicate that by proper modification of the first stage of Kinesthetic Teaching and incorporating the domain experience of the human teacher, we can remove the necessity of the second "reward" stage. One important point to be noted is that our approach puts considerable importance on the quantitative expertise obtained from the human teacher through assignment of different priorities to different shots. It is assumed that the human teacher can not only demonstrate a good shot movement, but also that he/she has enough expertise in that domain. It should be noted that changes in the experimental setup (for example: change in background color of the setup) would affect the learning process and hence would require recalibration of the parameters involved. Validation of the proposed method with the robotic manipulator is kept for our future work.

REFERENCES

Bentivegna, D., Atkeson, C. and Cheng, G. (2004). Learning tasks from observation and practice. *Robotics and Autonomous Systems.* Vol. 47:2-3, P. 163–169.

Billard A, Calinon S, Dillmann R and Schaal S (2008). *Robot Programming by Demonstration.* New York: Springer, pp. 1371–1394.

Bitzer S, Howard M and Vijayakumar S (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Flash T and Hogan N (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neurosciences.* 5: 1688–1703.

Giszter S, Moxon K, Rybak I and J C (2000). Neurobiological and neurorobotic approaches to control architectures for a humanoid motor system. *Intelligent Systems and their Applications*. 15: 64–69.

Guenter F, Hersch M, Calinon S and Billard A (2007). Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*. 21(13): 1521–1544.

Jetchev, N. and Toussaint, M. (2013). Fast Motion Planning from Experience: Trajectory Prediction for Speeding up Movement Generation. *Autonomous Robots*. Vol.34:1-2, p. 111-127.

Kober J, Mohler B and Peters J (2008). Learning perceptual coupling for motor primitives. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Kober, J.; Peters, J. (2011). Policy Search for Motor Primitives in Robotics. *Machine Learning.* 84, 1-2, pp.171-203.

Kormushev, P, Calinon, S, and D. Caldwell (2011). Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input. *Advanced Robotics.* pp. 1–20.

Mülling, K., Kober, J., Krömer, O., Peters, J. (2013). Learning to Select and Generalize Striking Movements in Robot Table Tennis. *International Journal of Robotics Research.* 32(3), pp. 280–298.

Peters J and Schaal S (2008a). Natural actor-critic. *Neurocomputing*. 71: 1180–1190.

Peters J and Schaal S (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks*. 21: 682–697.

Schaal S (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*. 6: 233–242.

Schaal S, Atkeson CG and Vijayakumar S (2002). Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*. 1: 49–60.