Recognition (along with Generation) of Actions from Demonstration

Mustafa Parlaktuna, Doruk Tunaoglu, Erol Sahin
Kovan Research Lab. Department of Computer Engineering Middle East Technical University, Ankara, Turkey

Emre Ugur
NICT, Brain ICT Labs, Kyoto, Japan
ATR, Cognitive Mechanisms Labs, Kyoto, Japan

***Abstract*— The studies on mirror neurons observed in monkeys indicate that recognition of other's actions activates neural circuits that are also responsible for generating the very same actions in the animal. The mirror neuron hypothesis argues that such an overlap between action generation and recognition can provide a shared worldview among individuals and be a key pillar for communication. Inspired by these findings, this paper extends a learning by demonstration method for online recognition of observed actions. The proposed method is shown to recognize and generate different reaching actions demonstrated by a human on a humanoid robot platform. Experiments show that the proposed method is robust to both occlusions during the observed actions as well as variances in the speed of the observed actions. The results are successfully demonstrated in an interactive game with the iCub humanoid robot platform.**

I. INTRODUCTION

Studies on mirror neurons [7] indicate that monkeys recognize other's actions using the very same neural circuits that are active during the generation of those actions by the monkey. The use of action generation circuits for recognizing other's actions (rather than using separate recognition circuits that are detached from action generation) allows the organism to link his own experiences with the ones observed in others, enabling the understanding of intentions.

Specifically, this paper adapts the Dynamical Movement Primitives (DMPs) [3], [8], [2], proposed as a learning by demonstration method, for online recognition (i.e. recognition before the completion of the observed action). The proposed method, called Closed Loop Primitives (CLPs), is shown to recognize and generate different reaching actions demonstrated by a human on a humanoid robot platform. Experiments show that the proposed method is robust to both occlusions during the observed actions as well as variances in the speed of the observed actions. The results are successfully demonstrated in an interactive game with the iCub humanoid robot platform.

In a prior study [1], we had laid out how DMPs can be extended to support online recognition. However these extensions had created convergence problems for generation, and the method was demonstrated for *only recognition*. This paper addressed these problems allowing us to *recognize as well as generate* actions on the robot from demonstrations.

II. SYSTEM ARCHITECTURE

In this section, we first explain Dynamical Movement Primitives (DMPs), then we describe our extension Closed Loop Primitives (CLPs).

A. Dynamical Movement Primitives (DMPs)

DMPs is a method proposed for learning behaviours from demonstrations. DMPs shown in equation 1, generate acceleration of an action for a given goal position, states of the robot(i.e. end-effector position and velocity).

$$\ddot{\mathbf{x}} = \underbrace{\mathbf{K}(\mathbf{g}-\mathbf{x}) - \mathbf{D}\dot{\mathbf{x}}}_{Linear\ Part} + \underbrace{\mathbf{K}\ \mathbf{f}(s,\mathbf{W})}_{Non-linear\ Part} \quad , \tag{1}$$

where $\mathbf{x}_o$ is the starting position, s is the phase variable, g is the goal position, $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$ are position, velocity, and acceleration vectors in task space respectively. $\mathbf{f}(\cdot)$ is a non-linear function and $\mathbf{W}$ contains the parameters for $\mathbf{f}(\cdot)$. $\mathbf{K}$ and $\mathbf{D}$ are diagonal matrices.

The phase variable s is defined as:

$$s_0 = 1 \tag{2}$$

$$\dot{s} = -\alpha s \tag{3}$$

$$\implies s = e^{-\alpha(t-t_0)} \tag{4}$$

It can be seen from equation 4 that s decays from 1 to 0 exponentially in time, decreasing the contribution of the non-linear part so the system converges to the goal position, and α is calculated by forcing $s = s_f$ at the end of an action.

DMP system in equation 1 can be analyzed in two parts: (1) Linear part which is inspired by a mass-spring-damper (MSD) system. (2) The non-linear part which perturbs the system to generate complex trajectories.

Learning by demonstration (LbD) refers to estimating action generation parameters given recorded trajectories of human demonstrations. It is done as follows: First, the states of the action $(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{g})$ are recorded from a single demonstration. Then, $\mathbf{f}(\cdot)$ is left alone and $\mathbf{f}(\cdot)$ values are calculated. Corresponding s values are calculated using equation 4. Finally, $\mathbf{f}(\cdot)$ with respect to s is learned using a learning algorithm (regression in the cited work).

In addition, DMPs do not have any online recognition scheme implemented in previous works although offline recognition is implemented in a previous work [4], and it is not suitable for online recognition since:

- The non-linear part of the DMPs runs in open loop since phase variable s is updated without feedback making the method vulnerable against perturbations.
- The phase variable s depends on the starting time of an action (equation 4) which may not be observed by the observer.

- α in equation 4 has to be calculated from the end of the motion which prevents online recognition.

B. Closed Loop Primitives (CLPs) - Modifying DMPs for Online Recognition

We propose Closed Loop Primitives (CLPs) equation 5 where we have modified the DMP formulation [2] for online recognition.

$$\ddot{\mathbf{x}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\dot{\mathbf{x}} + w_f \ \mathbf{K} \ \mathbf{f}(\mathbf{z}, \mathbf{W}), \tag{5}$$

where w_f is the weighing factor, $\mathbf{z}$ is the state vector and $\mathbf{W}$ is the parameters of the function approximator $\mathbf{f}(\cdot)$, other variables are same as DMPs. The state vector is added to make the non-linear part closed loop, which is restricted to variables observable by others. To keep it simple, we used the current position and velocity of the end-effector as the variables of the state vector:

$$\mathbf{z} = [\mathbf{x} \ \ \dot{\mathbf{x}} \ ]. \tag{6}$$

The weighing factor w_f is defined to ensure convergence because by removing the phase variable s from DMPs we removed the quarantee for convergence:

$$w_f = \exp(-\int h(\cdot) \ dt), \tag{7}$$

where $h(\cdot)$ is the instantaneous penalty which penalizes movements away from goal position. w_f starts from 1 and decays to 0 as the instantaneous penalties accumulate. $h(\cdot)$ has to be chosen appropriately so that the non-linear part is penalyzed whenever the system diverges from the goal:

$$h(\dot{r}) = p_0 \ \exp(\ p_1 \ \exp(\ p_2 \ \dot{r} \ )), \tag{8}$$

$$\dot{r} = \frac{\dot{\mathbf{x}} \cdot (\mathbf{x} - \mathbf{g})}{||\mathbf{x} - \mathbf{g}||}, \tag{9}$$

where $\dot{r}$ is the radial velocity of the end-effector with respect to the goal (i.e. change in the distance between current position and the goal) and p_0, p_1 and p_2 are the parameters of the Gompertz function which defines a sigmoid function proposed by Benjamin Gompertz where p_0 sets the upper asymptote, p_1 sets the displacement in x axis and p_2 sets the growth rate.

CLPs are guaranteed to converge to the goal point since the non-linear part will diminish weight whenever the end-effector gets away from the goal. Furthermore, CLPs run in a completely closed loop fashion since observable variables are used and since there is no explicit time variable.

C. Learning by Demonstration

In CLPs, learning by demonstration corresponds to learning $\mathbf{f}(\cdot)$ (estimating its parameters $\mathbf{W}$) given the time evolution of $\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}$ and the goal point $\mathbf{g}$ recorded from multiple demonstrations. One $\mathbf{f}(\cdot)$ is learned for each action irrespective of the goal position meaning each action can be actuated on any object. As a result one CLP is learned from multiple demonstrations for each action.

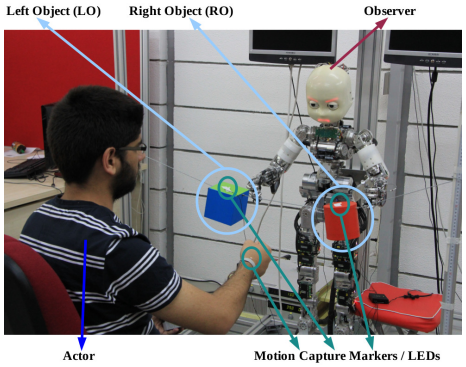


Fig. 1. Experimental Setup. Motion Capture Tracker is attached to the ceiling (not shown).

$\mathbf{f}(\cdot)$ can be observed variables, as calculated in equation 5:

$$\mathbf{f} = \frac{\mathbf{K}^{-1}\left[\ddot{\mathbf{x}} - \mathbf{K}(\mathbf{g} - \mathbf{x}) + \mathbf{D}\dot{\mathbf{x}}\right]}{w_f} \tag{10}$$

where w_f is calculated from equation 7 using $\mathbf{g}$ and $\mathbf{x}$. The last point of the trajectory is used as the goal point.

To acquire a new behavior from demonstrations, the mapping from $\mathbf{z}$ to $\mathbf{f}(\cdot)$ has to be learned. Given multiple demonstrations, first each trajectory is translated to a goal centered frame by subtracting the goal position from all positions. Then, $\mathbf{f}(\cdot)$ is calculated using equation 10. The final step is to estimate the parameters of the non-linear part ($\mathbf{W}$) using samples of $\mathbf{z}$ as inputs and the corresponding samples of $\mathbf{f}(\cdot)$ as outputs. We used Vijayakumar's implementation of LWPRs [9] to learn the relation between $\mathbf{z}$ and $\mathbf{f}(\cdot)$ using the default parameters.

D. Online Recognition

During recognition, initial observation of the state vector ($\mathbf{z}_0$) is used to calculate future trajectories for every possible action-object pair (goal positions are known by the observer) using the learned models. As the action unfolds, the observed trajectory is compared with the mentally simulated trajectories and cumulative error for each pair is calculated:

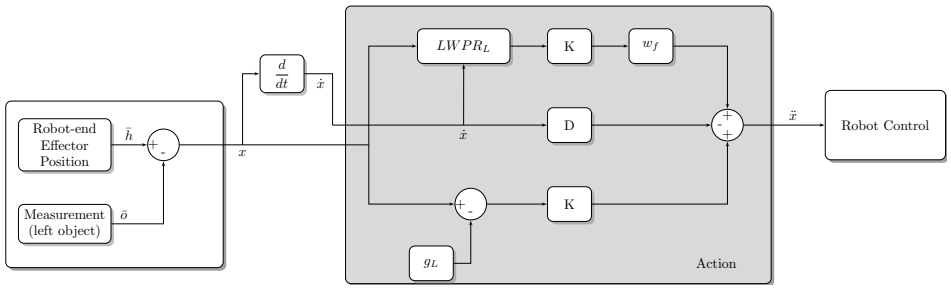
$$err_i(t_c) = \sum_{t=t_0}^{t_c} ||\hat{\mathbf{x}}_i(t) - \mathbf{x}|| \tag{11}$$

where err_i is the i^{th} pair's cumulative error, t_0 is the starting time, t_c is the current time, $\hat{\mathbf{x}}$ is the i^{th} pair's simulated/predicted position and $\mathbf{x}$ is the observed position. Note that positions are calculated in object centered frame and thus different objects will yield different $\hat{\mathbf{x}}_i(t)$ and $\mathbf{x}(t)$ values.

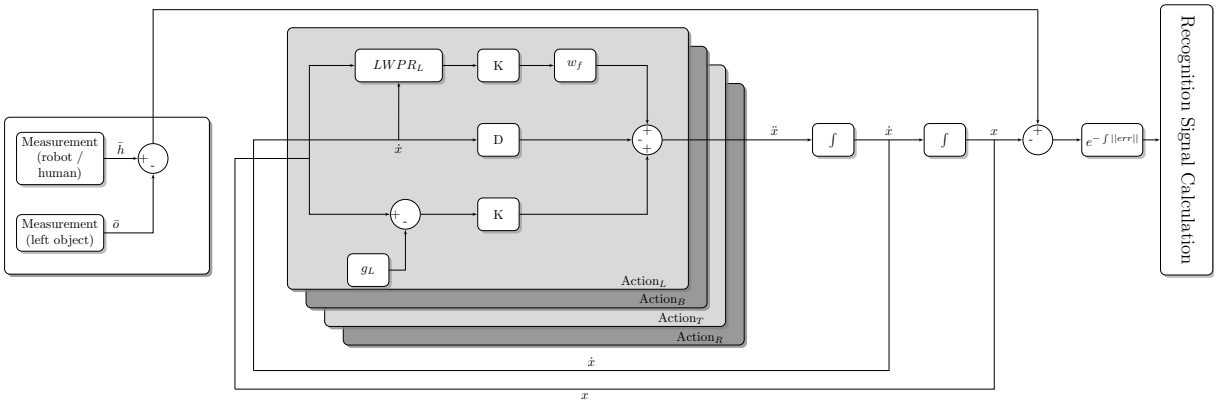
In order to have a quantitative mesasure of similarity between actions, we define recognition signals as:

$$rs_i(t_c) = \frac{e^{-err_i(t_c)}}{\sum_j e^{-err_j(t_c)}}, \tag{12}$$

where rs_i is high for pairs with low error. A recognition decision is made if the ratio of the two highest recognition signals gets above a certain threshold (Υ).



(a) Generation



(b) Recognition

Fig. 2. (a): Cartesian state of the end-effector is tracked and translated to a different object centered frame for each object. When recognition starts, the observed state is used as the initial state to the learned behaviors(not shown). Then the learned LWPR model of each action is used to simulate future trajectories. As the action unfolds, observed and simulated trajectories are compared and responsibility signals are calculated. From these signals, recognition decision is made according to the threshold. Note that action choice determines the used LWPR model while the object choice determines the input of the LWPR model. (b): Cartesian state of the end-effector is obtained from the robot and translated to a object centerd frame for each object. When the robot decides to act, the LWPR model of the decided action is used similar to the recognition architecture. The generated acceleration command is sent to the motors.

III. Experimental Setup

The experimental setup consists of two objects, an actor, an observer (the robot) and a motion capture system (Figure 1). The VisualEyez^{TM}VZ 4000[1] motion capture system which records the 3D Cartesian positions of markers at 100Hz is used for tracking targets. One marker is attached to the right wrist of the actor (end-effector) and one marker is attached to the center of each object.

A. Actions

Four actions are defined as:

- **L(X):** Reaching X from left.
- **R(X):** Reaching X from right.
- **T(X):** Reaching X from top side.
- **B(X):** Reaching X from bottom side.

and applied on two objects, left object (LO) and right object(RO).

The flow diagram of the recognition architecture for reaching to left on left object is depicted in figure 2(b) and the flow diagram of the generation architecture reaching to left of the left object is shown in figure 2(a).

[1]http://www.ptiphoenix.com/VZmodels.php

B. Training & Test Sets

Using the target tracking system, different actions on both objects are recorded. Beginning and end parts of each recording is truncated by a speed threshold to discard motionless parts (threshold = 0.05m/s).

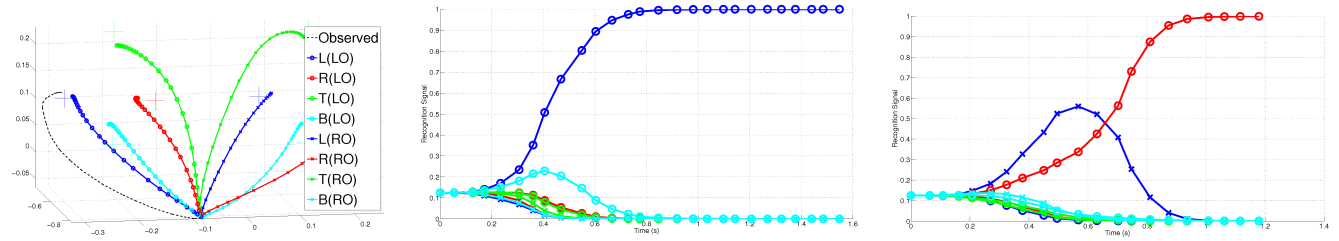
To form the training set, a 3D workspace is defined for the starting points for actions. The size of the workspace is maximized by selecting closest and furthest points where the actions can be performed comfortably[2]. $3 \times 3 \times 3$ (3 points for each dimension) are selected for the workspace which results in 27 points in total. From each starting point 2 repetitions for each object are performed which results in 108 recordings per action.

Test set is composed of 25 trajectories for each object-action pair. The actor has selected initial positions randomly while remaining in the 3D workspace defined for the training set.

C. Learning

In order to learn from demonstrated actions, the state vector ($\mathbf{z}$) and the corresponding $\mathbf{f}(\cdot)$ are calculated for each sample in each trajectory using the training set. Then, an

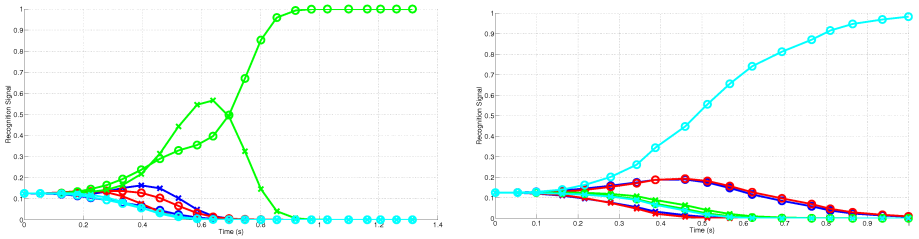
[2]$\pm$30 cm in left/right and up/down directions, 20-60 cm outwards with respect to the midpoint of object centers.



(a) Observed action and mentally simulated trajectories L(LO)

(b) Recognized action L(LO)

(c) Recognized action R(LO)



(d) Recognized action T(LO)

(e) Recognized action B(LO)

Fig. 3. Observed trajectory shown with black line and the corresponding recognition signals. Axes for trajectories are in meters.

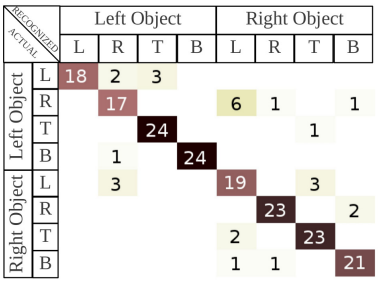


Fig. 4. Confusion matrix for $\Upsilon = 4$.

LWPR model is trained for each action until the performance stops increasing for 10 epochs (maximum 100 epochs). This procedure takes approximately 4 minutes for each action on a PC with a CPU running at 2.67 GHz and a RAM of 4GB.

D. Goal Estimation

The goal positions for each action (i.e. offsets with respect to the object center) are calculated by taking the average of the goal position for each of the two demonstrations in the training set. For a single demonstration the goal position is taken as the last position in the trajectory in the reference frame of the acted object.

IV. Recognition

We evaluate the recognition performance of CLPs as follows:

- When the beginning of an action is observed, mental simulation of trajectories for each action-object pair is initiated.
- As the action unfolds, mentally simulated trajectories are compared with the observed trajectory.
- Recognition signals (RS) are calculated from the errors where a high signal corresponds to a low error.
- An action-object pair is chosen as the recognized action if largest RSs ratio to the second largest RS gets above the threshold Υ.

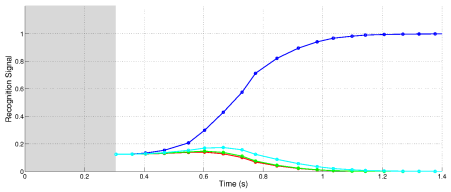
Figure 3(a) shows an observed trajectory and 8 mentally simulated trajectories. Recognition signals are shown in figures 3(b), 3(c), 3(d), 3(e) for different demonstrations from the test set. Although the system may make a mistake at the beginning as in figures 3(c) and 3(d), the recognition signal of the correct action triumphs over others as the action unfolds.

We calculated the confusion matrix for $\Upsilon = 4$ (using position comparison) which gives 85% success rate where an action is recognized before 50% is completed on average. Figure 4 shows that greatest confusion is between R(LO) and L(RO) which is an expected result since the beginning part of the trajectories are similar for these actions although their velocity profile is different. Furthermore, some of the rows do not sum up to 25 which shows that some actions are left unrecognized because of the threshold.

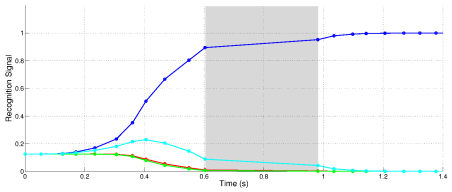
V. Recognition under Occlusion

In this experiment, we focused on three scenarios:

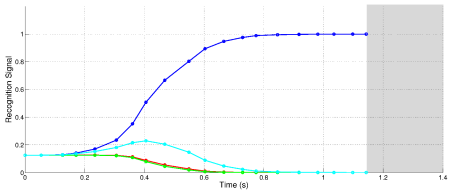
- **Occlusion in the begining**: The beginning of an action is occluded and mental simulation starts when the occluded part ends and the first sample is observed.
- **Occlusion in the middle**: The middle of an action is occluded. Mental simulation starts with the first sample and continues until the end, however errors and RSs are not updated for the occluded part since there is no observation there.
- **Occlusion in the end**: The end of the action is occluded. Mental simulation and recognition signal calculation is done as usual but ends in the occluded part. If a decision is not yet made, then the observation is left undecided.



(a) Blocked at beginning



(b) Blocked at middle



(c) Blocked at end

Fig. 5. Recognition signals with respect to time when 20% of the observation is occluded for all 3 scenarios

First, we calculated the recognition performance for the above scenarios for $\Upsilon = 4$. Then, we systematically analyzed the effect of threshold values for different occlusion percentages in order to find the best success rate and the corresponding threshold value. We did this because an RS may not dominate above others when the action is occluded, causing it to be left undecided.

Figure 5 shows the recognition signals when 20% of the action is occluded. Figure 5(a) shows when occlusion is at the beginning. As seen from the plot the system is able to cope with missing the beginning of an action. Figure 5(b) shows when occlusion is at the middle, and figure 5(c) shows when occlusion is at the end of an action. The recognition signals does not change during occlusion since system does not update itself.

In figure 6, success rates for $\Upsilon = 4$ for the three scenarios with respect to the percentage of the blocked part is shown. As seen from the figure, success rates are high for all cases for considerable amounts of occlusion ($\approx$40%). Furthermore when the beginning of an action is occluded, small amount of occlusion (upto 20%) gives better results.

Figure 7 shows the threshold values that give the best results for different amount of occlusions. As the amount of occlusion increases, smaller thresholds give better results since the number of undecided cases decreases.

The success rates when the middle of an action is occluded are unexpectedly high, especially for lower thresholds. This is mostly due to the continuation of mental simulation during occlusion without updating recognition signals: Although decision times are not shown in the figure, most actions are decided after the occluded part and as the amount of occlusion increases decisions are made closer to the final point of an observation, i.e. as if decided after the action is completed.

Results show that our system can cope with considerable amount of occlusion whether the occluded part is in the beginning, middle or end. Furthermore, it can be suggested: (1) To use a higher velocity threshold to assume that an action is started and to start mental simulation, since small amount of occlusion gives better results for occlusion in the beginning, (2) to use an adaptive threshold when an occlusion is detected since smaller thresholds gives better results when the amount of occlusion increases, and (3) to force the system to decide on the best option when the amount of occlusion is very high since threshold values close to one gives the best result for these cases.

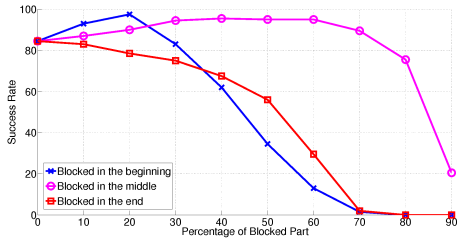


Fig. 6. Success rates versus the percentage of occlusion for $\Upsilon = 4$.

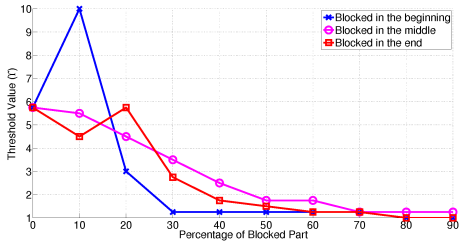


Fig. 7. Thresholds that give the best results.

VI. Recognition under Different Speeds

In this experiment we tested the recognition performance of the system when the same action is observed with different speeds. Since it is hard to record the same action with different speeds, we modified the time stamps of an action while leaving the observed positions as they are. Through

this, an action can be replayed at different speeds without changing the position profile.

From figure 8, it can be seen that our system obtains a 60% success rate even when the speed of an action is doubled or decreased to 70%. Using different thresholds for different speeds is not a feasible approach since $\Upsilon = 4$ gives the best result for most of the cases.

This experiments show that our system can cope with speed changes in the range of $0.7X - 2X$ while obtaining success rates higher than 60%. When comparing two trajectories, our algorithm compares the points with closest time stamps.

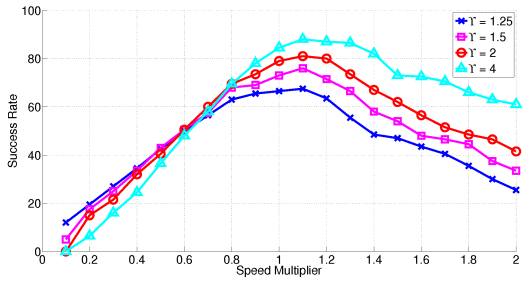


Fig. 8. Success rate vs. the speed multiplier for different thresholds.

VII. Robot Demonstration

We used the proposed system in an interactive game that can be played between a human actor and humanoid robot iCub [5].

The scenario of the game is as follows: The actor performs one of the 8 behaviors (2 objects, 4 actions). iCub blinks when it recognizes an action and reacts with the appropriate counter action in order to bring his hand to the opposite side of the acted object. Then iCub returns to its steady state when the actor brings its hand away from the object and the cycle starts from the beginning.

It should be noted that iCub uses the same system (LWPR model) while recognizing an action and generating the same action. The system runs with a loop period of 10ms (100Hz) both during the recognition phase and the generation phase. For converting velocity commands in Cartesian space to joint velocities we have used the iKin library [6] which can produce smooth human like motions. Although CLPs generate acceleration commands, iKin and most of the other inverse kinematic libraries does not support acceleration commands and thus we have integrated the acceleration commands to obtain velocity commands before sending them to the inverse kinematic library.

Video of this interactive game can be found in our website[3] and snapshots of the video can be seen in figure 9.

VIII. Conclusion

In this paper, we described CLPs, a method for learning, generation and recognizing actions, which is developed by modifying DMPs. CLPs are shown to generate and recognize

[3]http://www.kovan.ceng.metu.edu.tr/pub/video/recognitionLow.mp4

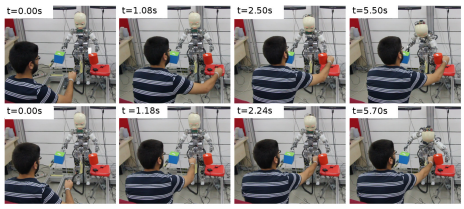


Fig. 9. Interactive game with the robot. Each row shows a different demonstration. First column shows the starting point of the actions. Second column shows the moment where iCub recognizes the actions (indicated by the eye blink). Third column shows the end of the actions for the actor and the last column shows the point where iCub finishes its actions.

different reaching actions online, using the same architecture similar to mirror neurons. Also experiments show that CLPs are robust to occlusions during observed actions, and speed variances of the observed actions.

The system was demonstrated as an interactive game with iCub humanoid robot, showing that the system is able to recognize and generate actions against real life problems.

Acknowledgements

This work was founded by the European Commission under the ROSSI project (FP7-ICT-21625) and the TÜBİTAK through project 109E033.

References

[1] B. Akgün, D. Tunaoğlu, and E. Şahin. Action recognition through an action generation mechanism. In *10th International Conference on Epigenetic Robotics*, 2010.

[2] H. Hoffman, P. Pastor, D., and S. Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, pages 2587–2592, May. 2009.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 752 –757 vol.2, 2001.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1547–1554. MIT-Press, 2003.

[5] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The icub cognitive humanoid robot: An open-system research platform for enactive cognition. In *50 Years of Artificial Intelligence*, pages 358–369. Springer Berlin / Heidelberg, 2007.

[6] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1668–1674, Oct. 2010.

[7] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive brain research*, 3:131–141, 1996.

[8] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research*. Springer, 2003.

[9] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005.