Action Effect Generalization, Recognition and Execution through Continuous Goal-Directed Actions

Santiago Morante, Juan G. Victores, Alberto Jardón and Carlos Balaguer

Abstract— **Programming by demonstration (PbD) allows matching the kinematic movements of a robot with those of a human. The presented Continuous Goal-Directed Actions (CGDA) is able to additionally encode the effects of a demonstrated action, which are not encoded in PbD. CGDA allows generalization, recognition and execution of action effects on the environment. In addition to analyzing kinematic parameters (joint positions/velocities, etc.), CGDA focuses on changes produced on the object due to an action (spatial, color, shape, etc.). By tracking object features during action execution, we create a trajectory in an n-dimensional feature space that represents object temporal states. Discretized action repetitions provide us with a cloud of points. Action generalization is accomplished by extracting the average point of each sequential temporal interval of the point cloud. These points are interpolated using Radial Basis Functions, obtaining a generalized multidimensional object feature trajectory. Action recognition is performed by comparing the trajectory of a query sample with the generalizations. The trajectories discrepancy score is obtained by using Dynamic Time Warping (DTW). Robot joint trajectories for execution are computed in a simulator through evolutionary computation. Object features are extracted from sensors, and each evolutionary individual fitness is measured using DTW, comparing the simulated action with the generalization.**

I. Introduction

Humans are able to easily extract the main consequences of an action performed on an object. However, in usual robot imitation, there is a lack of codification of action effects, and only the kinematic aspects are considered (it is a kind of blind imitation). This fact limits flexibility in action execution. We aim to solve this problem by incorporating a way to encode the consequences of an action.

Literature from psychology indicates that the human brain encodes actions as end-goals. For example, when children imitate others grasping a person's ear, they tend to imitate the action goal (which ear to grasp) rather than the kinematic aspects of the action (which hand is used to perform the grasping) [1].

Neuroscience has also discovered some points supporting goal coding of actions, especially in the study of *mirror* neurons [2]. Mirror neurons fire both when an action is performed and when the same action performed by another subject is observed. As shown in [3], when monkeys were trained to grasp food with a tool, their mirror neurons activated when they performed the action, and also when observing the action performed by an experimenter using a different tool. This demonstrates their kinematic independence.

All of the authors are members of the Robotics Lab research group within the Department of Systems Engineering and Automation, Universidad Carlos III de Madrid (UC3M). smorante@ing.uc3m.es

Inspired by this neuronal behavior, we present Continuous Goal-Directed Actions (CGDA) and define them as actions in which the only parameters analyzed are the ones belonging to the object (or more generally, elements) affected by the action in a continuous time. We define two kinds of goal directed actions:

- Goal only: When the information used is only the difference between the initial and the final state of the element. This was studied in [4].
- Continuous tracking: When the whole process of change is taken into account. This paper focuses on this kind of goal directed actions.

We consider Continuous Goal-Directed Actions a useful way to teach robots the consequences of an action. Robots are usually taught how to perform a task by imitating how humans do it, guiding its elements through an action execution. The robot can, effectively, repeat the same movements, but the action goals remain a mystery for it [5]. By using CGDA, an action can be encoded in a complementary way, by learning, not only how to do it, but also the effects of the action. This *double learning* may allow the robot to complete the action, even when the scenario changes or when elements block its usual path of execution, by generating alternative motions which accomplish the encoded goals.

II. Related Work

Robot imitation is usually denoted as programming by demonstration (PbD) [6]. The way these methods generalize an action is by recording the kinematics of a demonstrator when performing the action, and then applying different machine learning algorithms. The demonstrator can either be the guided robot itself, a human with sensors attached, or video sequences of human movements.

In [6], a human demonstrator performs a task several times (e.g. hitting a ball) using a robotic arm. Positions, orientations and velocities of the arm are recorded, and the number of representative states of the action are estimated with Hidden Markov Models (HMM). HMM are used to handle spatio-temporal variabilities of trajectories across several demonstrations. Finally, and in order for the robot to execute the trajectory, Gaussian Mixture Regression (GMR) is used to create a regression function using previous HMM states. This reconstructed trajectory is the one the robot reproduces to imitate the human movement. Another common technique used, along with HMM [7] [8], is Gaussian Mixture Models (GMM) as in [9] [10].

Recognizing an action through external measurements is called direct action recognition. In [11], they perform a neuro-fuzzy classification of optical flow features between consecutive frames of human movement in video sequences. Neuro-fuzzy is a combination of fuzzy logic with neural networks, using the classified output of a fuzzy system as an input to the neural network. In [12], they track and filter human hand and feet trajectories through Principal Component Analysis (PCA). First, they record trajectories of key points from a video. Then, they split them into sub-units called basic motions. Next, they extract some features of the basic motions, and project these feature vectors into a reduced space generated by PCA, resulting in the formation of clusters of similar actions. For recognition purposes, they record an action, transform it with the same process explained, project its vector onto the reduced space, and finally, associate it with the closest cluster.

As shown, the focus in these types of research is on learning the kinematics of actions. By using only kinematics, actions are limited to be executed exactly as taught. Any disturbance, e.g. a blocking path or a displaced element, would make the task completion impossible. This is why a complementary effect encoding is also important.

When talking about goal directed actions in robotics, a goal encoding is found in [13] where, despite they learn the kinematic trajectory to perform an action, they also encode some goals to be achieved. They replicate a psychological experiment [1] with children, where, in a table, there are colored dots which are touched by a human with both arms in alternation. When the dots stay on the table, children tend to imitate the goal (what dot to touch), and not the arm used to do it. In the replicated experiment, the demonstrator repeats the same task and, while observing the demonstration, the robot tries to extract a set of constraints for the task. Later, the robot computes the trajectory that best satisfies the constraints and generates a motion.

There are no exclusively continuous tracking goal directed actions references in literature, to the authors knowledge. The only slightly related one found [14] uses a combination of object spatial and demonstrator-hand movement tracking. In [14], they build a system with a set of primitive actions (inverse models). When the human demonstrator performs an action, they continuously track the object and the hand spatially through time. At the same time, they run all inverse models during action stages to find the best performance of each model in each stage. Finally, they construct a high-level inverse model composed by those selected primitives, being able to imitate the action goal with similar spatial movements. Notice that the parameters to be imitated are not robot joint positions, the only target here is the hand position. The object tracking is only used to identify grasping and releasing stages. Despite the continuous tracking used in this work, they do not fully exploit the benefits of the object features variation.

Our work aims to allow a complete CGDA infrastructure, using features beyond spatial ones, such as changes in color or object deformations.

III. Continuous Goal-Directed Actions

Continuous Goal-Directed Actions are a way to encode the effects of an action when the action is demonstrated to a robot. The main differences between the CGDA and PbD paradigms can be found in Table I. We have developed a continuous tracking infrastructure to allow the learning of actions with relevant object feature intermediate states e.g. recognizing the rotation of a valve is unachievable without a continuous tracking infrastructure, because the final state of the valve could be the same as the initial, looking like no action has been executed.

We present a method to generalize, recognize and execute actions by its effects on objects, in a continuous tracking way. This tracking produces a trajectory in a n-dimensional feature space (where n equals the number of tracked object features). When trajectories are considered in a discrete way, the action repetition creates a point cloud in time. Throughout the rest of the paper, we will refer to *action trajectory* as the object feature trajectory.

A. Generalization

For generalization purposes, we need to extract a representative n-dimensional trajectory of the point cloud from several repetitions. This process is composed by the following three steps.

1) Time Rescaling: Before inserting a single action repetition in the point cloud, it is normalized in time (range $[0, 1]$). With this time rescaling, every action execution gets bounded by the same temporal limits, making the algorithm independent of the repetitions speed. All normalized trajectories are introduced in the same object feature space, forming a point cloud.

2) Average in Temporal Intervals: To model the point cloud, we split it in temporal intervals, fixing one interval per second. The number of seconds is computed from the average duration of the original repetitions. Each interval contains points of all repetitions, in the same percentage of execution. As the repetitions are normalized in time, each interval represents a percentage of action execution, and the number of intervals allows preserving a notion of the action duration. The representative point of each interval is extracted as the average for each dimension of all points of the interval, as seen in Fig. 1. These temporal averaged points are the ones used in the posterior interpolation.

3) Radial Basis Function Interpolation: Once we have the representative points of the point cloud, we have to join them to form a generalized action, i.e. an object feature trajectory we can consider as a generalization. In a robot joint space, an interpolation could create a jerky joint trajectory, so literature, e.g. [6], commonly uses regressors such as GMR. However, working in the object feature space, we use an interpolator to assure the trajectories pass through the target points (which are the states of the object in an instant). We use a Radial Basis Function, which is an interpolation technique based on measuring the influence of every known point over the queried point [15]. The RBF interpolation

TABLE I
MAIN DIFFERENCES BETWEEN PbD AND CGDA PARADIGMS.

	PbD	CGDA
Objective of imitation	Spatial trajectories	Object feature states
Features tracked	Demonstrator's joint positions/velocities	Object's shape, area, color, coordinates, etc.
Strengths	Perfect kinematic imitation	Effects encoding
Weaknesses	Undefined goal to achieve	Undefined way to achieve the goal

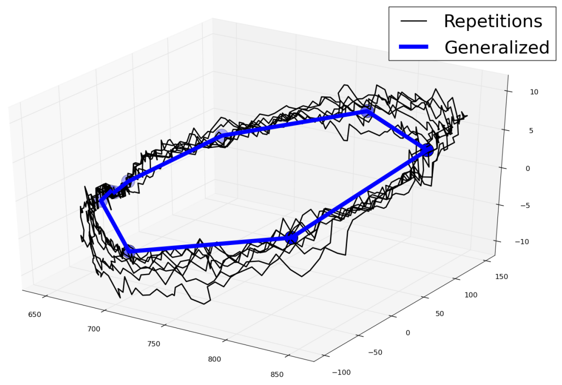

Fig. 1. Plot representing a three feature trajectory. Black lines are training action repetitions. The blue line is the generalization of all the repetitions.

$f(x)$, which will become the final generalized trajectory, is mathematically expressed as a sum of radial basis functions:

$$f(x) = \sum_{i=1}^{N} w_i \, \phi(\|x - x_i\|) \tag{1}$$

Where N is the number of radial basis functions, equal to the number of intervals, and x_i represents the coordinates of each interval's known point. The radial basis function is denoted as ϕ, where the input parameter is the distance between the known point x_i and the queried point x, measured with L^2 norm. The coefficient w_i is the weight of a specific known point over the queried point x, and it is the value to be solved. As the interpolation is known at known points, the weight problem is solved as a set of N linear equation with N unknowns:

$$\begin{aligned} f(x_1) &= \sum_{i=1}^{N} w_i \, \phi(\|x_1 - x_i\|) \\ &\vdots \\ f(x_N) &= \sum_{i=1}^{N} w_i \, \phi(\|x_N - x_i\|) \end{aligned} \tag{2}$$

From the available radial basis functions, we have selected the linear one (as we do not care on trajectory smoothness on the feature space):

$$\phi(r) = r \tag{3}$$

Where $r = \|x - x_i\|$. Once the interpolated function is returned, we consider this output as the generalized function of an action. Its physical meaning is how the state of the object, regarding its features, is changing across the action performance.

B. Recognition

Here we aim to recognize an action by comparing a query action with the previously generalized ones. We assume the recognition as the comparison of a query with our generalized trajectories, returning the one with the highest similarity.

The initial treatment of the query action is the same as explained. As they are normalized in time, we can take t values along time for each action to compare them. The technique used in the comparison is Dynamic Time Warping (DTW). DTW is an algorithm usually used to optimally align two temporal sequences [16]. Our use of DTW is to compare two time-dependent sequences of points $X = \{x_1, .., x_N\}$ and $Y = \{y_1, .., y_M\}$ with $N, M \in \mathbb{N}$. To compare two elements, a local cost measure (a distance $d(x, y)$) is needed. A lower cost represents a bigger similarity of the sequences.

Evaluating all pairs of points between the sequences, using in this case a L^2 norm, we obtain a cost matrix CM, with a size of $N \times M$:

$$CM = \begin{pmatrix} d(x_0, y_0) & \cdots & d(x_N, y_0) \\ \vdots & \vdots & \vdots \\ d(x_0, y_M) & \cdots & d(x_N, y_M) \end{pmatrix} \tag{4}$$

Once having this matrix, the goal is to find the lowest cost alignment path, which intuitively should run along the lowest cost cells. This alignment is called the warping path. DTW includes some constraints in the path calculation to assure a monotonic advance of the path and to assure that the first elements as well as the last elements are connected to each other. The total path cost $C_P(X, Y)$ is calculated as the sum of the local costs C:

$$C_P(X, Y) = \sum_{l=1}^{L} C(x_{nl}, y_{ml}) \tag{5}$$

Where L is the length of the path. For programming reasons, the path is usually calculated in an accumulated cost matrix, where each cell represents the cost of the correspondent pair (x, y) plus the cost to reach this cell (see Fig. 2). In the accumulated matrix, the normalized cost $C_{P_{norm}}(X, Y)$ of the optimal path is expressed as:

$$C_{P_{norm}}(X, Y) = \frac{C(x_n, y_m)}{N + M} \tag{6}$$

In our case, we use this normalized cost of the optimal path as a measure of discrepancy between dimensions. As DTW is computed between two signals for one dimension only, we consider the total cost of alignment between two n-dimensional trajectories as the sum of the costs of the optimal paths of each dimension, obtaining a single score D:

$$D = \sum_{i=1}^{n} C_{P_{norm}}(X_i, Y_i) \tag{7}$$

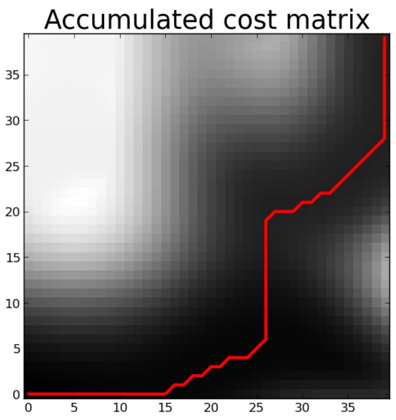

Fig. 2. Example of accumulated cost matrix for two sequences. White cells represent high cost, while dark cells are low cost ones. The red line is the lowest cost path.

This score is used as the measure of discrepancy between two trajectories in the n-dimensional space. In recognition, the trajectory with the smallest score is the one we consider the match.

IV. Experiments

Three different experiments have been performed. Two of them involve CGDA recognition and the last of them also includes CGDA execution.

A. Object Feature Trajectory Recognition

The first experiment framework consist in a Kinect camera pointed at a desktop tracking a colored marker. A demonstrator performs several action using the marker in front of the camera. Using YARP software [17], we connect the camera input with a computer vision library [18] to measure marker features. Four different basic actions were selected. We have given names to each basic action for simplicity in explanation, but semantics is not used in the process. The actions involve spatial movements (MOVE, ROTATE, WAX) and color changes (PAINT). Each action is described as follows:

- MOVE: Marker displacement of 30 cm in one straight direction.
- ROTATE: Rotation over the Center-of-Mass (CoM), on one axis, of 90 degrees.

TABLE II
DTW cost matrix: test actions (lower case) vs. generalized trajectories (upper case). Bold numbers represent best values (minimum discrepancy).

	MOVE	ROTATE	WAX	PAINT
move	**229**	332059	290334	552055
rotate	389021	**7606**	325211	694049
wax	402555	304669	**1724**	44259
paint	497152	671078	25896	**1277**

TABLE III
DTW cost matrix: reduced test actions (lower case) vs. reduced generalized trajectories (upper case). Bold numbers represent best values (minimum discrepancy). Only spatial features are used.

	MOVE	ROTATE	WAX	PAINT
move	**8.15**	10251.49	11428.03	4888.67
rotate	12836.77	**8.94**	10035.21	284.87
wax	12252.87	8977.23	**13.46**	5175.71
paint	4728.14	135.77	5021.54	**14.33**

- WAX: Keeping orientation fixed, movement of CoM over the perimeter of a circle of 30 cm of diameter (1 revolution).
- PAINT: Keeping its spatial coordinates fixed, the marker is painted in a different color with a marking pen, until almost all of the area is covered.

Seven repetitions of each basic action were recorded. Six of the repetitions were used to generate one generalized action. The final repetition of each set was used as a test action to be recognized. The tracked object features are: spatial location (X,Y,Z), area, HSV color (hue, value, saturation) and angle (of main axis). Each test action is passed to the CGDA recognition process, which compares it to each of the previously generated generalized actions. Results are shown in Table II.

To measure the influence of all dimensions (relevant or not) over the relevant ones, we perform the same comparison using only spatial features, and ignoring the rest. The results can be seen in Table III.

These results show that our initial assumption is correct, and the measure of discrepancy between similar feature trajectories is lower than between different actions. This makes the recognition algorithm to correctly associate all the test trajectories with their set. Tables II and III show the influence of additional dimensions on the comparison. As more dimensions are used, the quality of the results decays. Deeper analysis is needed to check how other comparisons techniques behave in this situation.

B. Cartesian Space Trajectory Recognition

To check whether the CGDA approach is useful, during the previous experiment, we also measure the Cartesian positions (X,Y,Z) of the human demonstrator's arm joints: hand tip, wrist, elbow and shoulder. The demonstrator did not care

TABLE IV

DTW COST MATRIX: CARTESIAN TEST ACTIONS (LOWER CASE) VS. CARTESIAN GENERALIZED TRAJECTORIES (UPPER CASE). BOLD NUMBERS REPRESENT BEST VALUES (MINIMUM DISCREPANCY).

	MOVE	ROTATE	WAX	PAINT
move	**0.0032**	0.1594	0.0448	0.1031
rotate	0.1563	**0.0123**	0.0939	0.0430
wax	0.0234	0.0323	**0.0003**	0.0371
paint	0.0486	0.0148	0.0300	**0.0004**

to perform the actions in a specific kinematic way, or even equal between repetitions, the only aim was to accomplished the, already explained, action description. When comparing test and generalized Cartesian actions (following the same scheme as previously), we obtain Table IV.

As shown in Table IV, in this case, the system also recognizes the actions correctly, but the score differences are lower. In Table II, the correct answer is 1 to 3 orders of magnitude lower, while in Table IV results are all quite similar. This proves that enabling CGDA, we are allowing the demonstrator to focus on task completion, rather than focusing on the kinematic consistency.

C. Robot CGDA Execution by Evolutionary Computation

Despite it is not the aim of the paper to establish a rigid method to perform an encoded CGDA, we have developed a simple framework to check its feasibility. We have used Evolutionary Computation (EC) and a simulated model of a humanoid robot [19], Fig. 3. We set a CGDA as the goal, with its generalized trajectory as the reference, and EC, starting from an initial random robot joint position trajectory, performs joint parameter evolution until convergence.

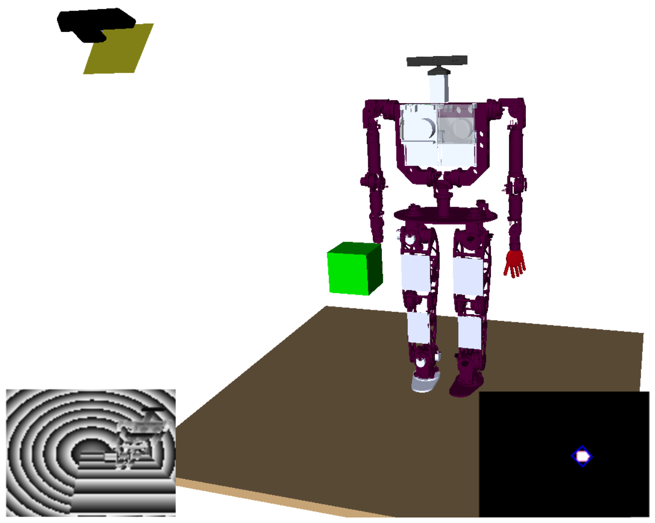

Fig. 3. The plot shows the experimental scenario with the robot, the object (green) and the Kinect camera. The bottom left square is the Kinect depth map and the bottom right square shows the color segmentation of the object.

EC performs a steady state selection algorithm (few individuals are replaced after the selection and crossover process). The following is a summary of the operator details:

1) Selection: A tournament is performed between 3 random individuals. Their fitness values are compared and winners are selected for crossover.
2) Crossover: Selection winners are crossed and their offspring substitute the worst individuals from the previous tournament.
3) Mutation: Finally, each child may be mutated with a 70% of probability.

The termination condition is set to 10 number of generations without improvement in the fitness value. The fitness evolution of our experiment can be found in Fig. 4.

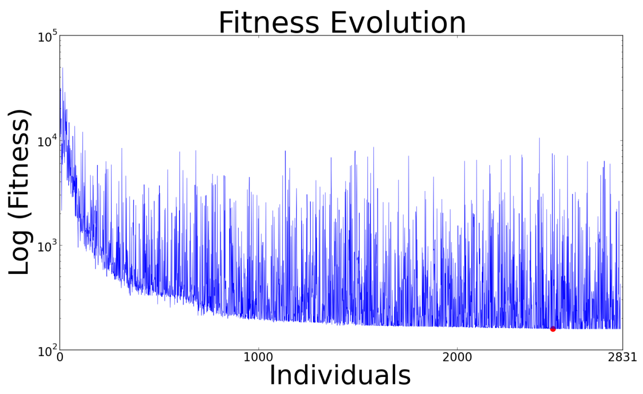

Fig. 4. Fitness value through evolution. The red point is the minimum value achieved by evolution.

EC evolves 30 values (3 joint positions, times 10 timesteps). Fitness is evaluated when the joint position trajectory is executed, by analyzing object features using a simulated Kinect camera in the environment. The reference action and the measured one are compared using CGDA recognition, and the score of discrepancy is used as the fitness value to minimize. We have chosen WAX as the action to be executed, and the object features measured are purely spatial ones (X, Y, Z). After convergence, the winner action is executed. Its performance compared to the generalized reference is depicted in Fig. 5.

The resulting trajectories, when executed on the robot, are not human natural movements. This is an expected behavior, as the aim of the experiment was to replicate object states during execution, and not the performance of human-like movements. In subsequent works we will focus on improving the precision of the execution.

V. CONCLUSIONS

One future line of research could be the integration of the presented *Continuous Goal-Directed Actions* framework with the *Programming by Demonstration* paradigm. A successful integration would require having the robot decide which features (object or human) are most relevant for recognizing or executing an action.

Another work to be developed in the future is parametric actions. Actions which depend on complements to be correctly performed are, at the current state of development,

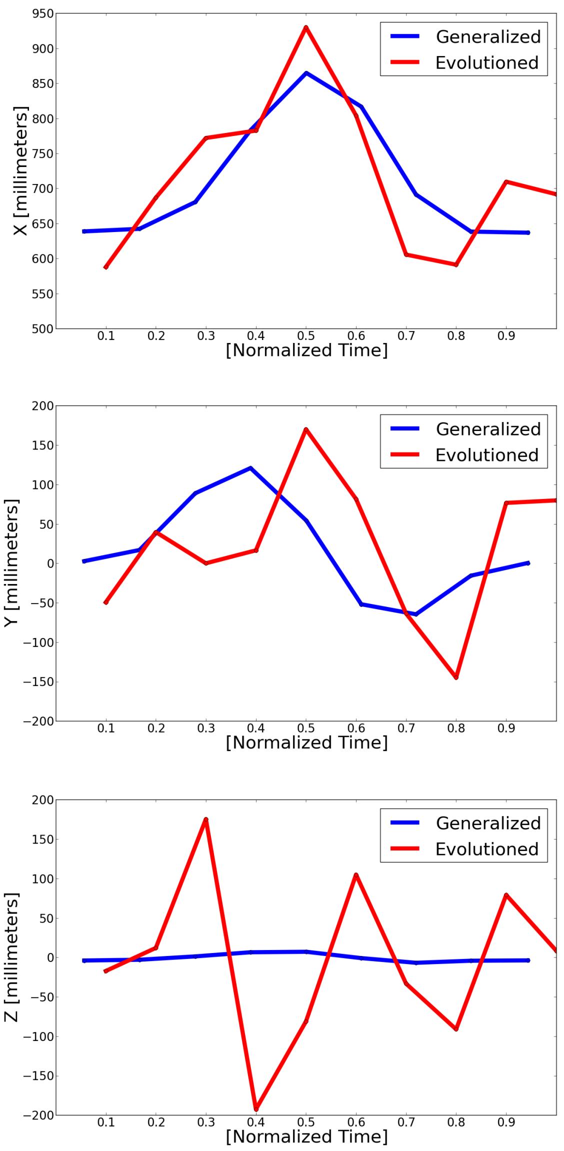

Fig. 5. Unidimensional temporal plots of generalized reference (blue), and the object feature space trajectory from executing the EC winner joint position trajectory (red). The Z dimension gives the worst results, the system was not able to reduce the error in this dimension.

not tackled. This is the case e.g. of the action *paint*: it is not the same to *paint blue* than to *paint red*. In our feature space, these actions would end in different coordinates for the *color* dimension. The solution to this problem could be to mix the structure here presented with semantic information, in a similar way to our previous work with objects [20].

Acknowledgments

Thanks to ARCADIA project DPI2010-21047-C02-01, funded by CICYT project grant, and RoboCity2030-II-CM project (S2009/DPI-1559), funded by Programas de Actividades I+D in Comunidad de Madrid and co-funded by Structural Funds of the EU.

References

[1] Harold Bekkering, Andreas Wohlschlager, and Merideth Gattis. Imitation of gestures in children is goal-directed. *The Quarterly Journal of Experimental Psychology: Section A*, 53(1):153–164, 2000.

[2] Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi, and Giacomo Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119(2):593–609, 1996.

[3] Magali J Rochat, Fausto Caruana, Ahmad Jezzini, Ludovic Escola, Irakli Intskirveli, Franck Grammont, Vittorio Gallese, Giacomo Rizzolatti, and Maria Alessandra Umiltà. Responses of mirror neurons in area F5 to hand and tool grasping observation. *Experimental brain research*, 204(4):605–16, August 2010.

[4] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. Semantic action parameter inference through machine learning methods. In *Robocity2030 12th Workshop Robotica Cognitiva*, 2013.

[5] Chrystopher L Nehaniv and Kerstin Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In *Interdisciplinary Approaches to Robot Learning*, volume 24, page 136. World Scientific, 1999.

[6] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *Robotics & Automation Magazine, IEEE*, 17(2):44–54, 2010.

[7] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2769–2774. IEEE, 2004.

[8] Sylvain Calinon and Aude Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of the 22nd international conference on Machine learning*, pages 105–112. ACM, 2005.

[9] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM, 2007.

[10] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[11] K Subramanian and S Suresh. Human action recognition using meta-cognitive neuro-fuzzy inference system. *International journal of neural systems*, 22(06), 2012.

[12] Daniel Stephen Chivers. *Human Action Recognition by Principal Component Analysis of Motion Curves*. PhD thesis, Wright State University, 2012.

[13] Sylvain Calinon, Florent Guenter, and Aude Billard. Goal-directed imitation in a humanoid robot. In *Proceedings of IEEE International Conference on Robotics and Automation, 2005 (ICRA 2005)*, pages 299–304. IEEE, 2005.

[14] Matthew Johnson and Yiannis Demiris. Abstraction in recognition to solve the correspondence problem for robot imitation. In *Proceedings of Towards Autonomous Robotic Systems*, pages 63–70. Springer, 2004.

[15] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[16] Thomas Albrecht. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, 2009.

[17] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp: yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48, 2006.

[18] Santiago Morante. Interfaz y librería para visión artificial, navegación y seguimiento en robótica. Technical report, Universidad Carlos III de Madrid, Leganés, 2012.

[19] Santiago Martínez, Concepción Alicia Monje, Alberto Jardón, Paolo Pierro, Carlos Balaguer, and Delia Muñoz. Teo: Full-size humanoid robot design powered by a fuel cell system. *Cybernetics and Systems*, 43(3):163–180, 2012.

[20] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. Towards Robot Imagination Through Object Feature Inference. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*. IEEE, 2013.