On Using Guided Motor Primitives to Execute Continuous Goal-Directed Actions

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

***Abstract*— In this paper, we study how human-robot interaction can be beneficial on the Continuous Goal-Directed Actions (CGDA) framework. Specifically, a system for robot discovery of motor primitives from random human-guided movements has been developed. These guided motor primitives (GMP) are used as scaffolds to reproduce a goal-directed actions. CGDA encodes goals as the changes produced on object features (color, area, etc) due to actions. This paper focuses on using motor primitives extracted from human-guided random robot movements to execute these goal-directed actions. The human guides the robot joints in random movements, which are later divided in small segments. These segments are compared in terms of joint positions and selected to be diverse. To perform goal-directed actions, the robot must discover an adequate sequence of GMP. To discover these sequences we organize the primitives as a tree with incremental depths (where each node represents a primitive) and use a breadth-first search. In one of the experiments performed, the robot executes a task based on spatial object features. In the other experiment, the goal is to paint a wall by following a color feature trajectory.**

I. IMITATION AND MOTOR PRIMITIVES

The field of robot imitation has focused traditionally on motor parameter reproduction. This approach has been called *programming by demonstration* (PbD) or *learning from demonstration* (LfD) [1]. These methods encode an action by recording the joint motor parameters of a demonstrator (either human or robotic) when performing the action, and then applying different several techniques to extract a generalization for a later reproduction.

Some authors questioned whether this motor imitation alone is useful for intelligent robots. Schaal asked in [2] as an 'outstanding question' to be answered: *How can the intention of a demonstrated movement be recognized and converted to the imitator's goal?*. From a wider perspective, [3] summarized the three key concepts of imitation: 'what to imitate', 'how to imitate', and 'when to imitate'. The Continuous Goal-Directed Actions (CGDA) encoding of tasks aims to fulfil mainly the 'what to imitate' concept [4]. We believe that robot imitation could be improved, and some of the problems stated solved, by taking more into account the action consequences in the environment. These consequences will usually be the goals of the task ('paint' action modifies the *color* of the painted object).

Goal directed actions are interesting because they enable a robot configuration independent way to encode and execute actions. This will allow a system to avoid the correspondence problem (the difference in the kinematic model of the demonstrator and the learner) [5]. Some examples are found in the literature involving goal directed imitation. In [6] they replicate a children psychological experiment. The experimental setup consists in a table and colored dots on it. Dots are touched by a human with both arms in alternation. In the psychological experiment children tend to achieve the goal (what dot to touch), and not the arm used to do it. In the robotic experiment, during the demonstration, the robot tries to extract a set of invariant constants. Later, the robot computes the trajectory that best satisfies the constraints. In [7] an object must be grasped and then placed at one of two presented targets that have different heights. There is a bridge shaped obstacle in the path. Depending on the height of the bridge, the object must be grasped differently and through a different path. In [8] there is a learning phase where the robot generates motion (as a combination of motor primitives) with no specific purpose, and analyses the consequences (sensory effect) of its actions. After the learning phase, when an experimenter performs actions, the robot recognized the observed actions by observing the consequences and encoding them as combination of motor primitives.

Literature has provided insights on how the human brain may use motor primitives for performing complex actions [9]. In his influential paper [2], Schaal explores the area of movement primitive what Schaal defines as *sequences of action that accomplish a complete goal-directed behavior*. Our development is close to this definition, despite the related works in motor primitives are not always making enough focus on the word "sequences". Instead of generating a single movement primitive to encode complete temporal behaviors, we aim to split movements and create small basic motions, which may be able to complete the required task when combined sequentially. This is an attempt to answer other Schaal's 'outstanding questions': *Is there a basic set of primitives that can initialize imitation learning?. How complex are the most elementary primitives in this set?. How can new primitives be learned?. How is sequencing and the recognition of sequences of movement primitives accomplished?*.

Dynamical system Movement Primitives (DMP) were used in [10] and subsequent works (e.g. [11], [12]). In DMP, dynamical systems are used to encode a policy (a mapping between world state and actions which enables a robot to select an action based upon its current world state [13]). This policy serves to accomplish an action by obtaining a reusable parametrized single-primitive policy. Despite DMP have been successfully used for learning complex, but unitary, tasks (like learning single primitive actions [14] [15]),

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

sequential combination of DMP has not yet been studied. The aim in [16] is similar to ours. They aim to acquire a library of primitives and select among them to adapt to new situations. For this adaptation they use primitive-associated parameters (called augmented state) which describe the task context from which the primitive was extracted. Instead of a sequential execution of primitives, they use an algorithm for applying a single action from a set of weighted primitives. For selecting the weights they use the augmented state. The demonstration set is obtained by a human guiding the robot to perform the desired tasks. Differently, we aim to obtain task-independent reusable primitives for task execution.

A developmental approach for self-discovery of primitives is found in [17]. In a similar way to [16], they encode primitives with descriptors, which describe situational parameters. For self-discovering motor primitives, they first execute several actions which randomly combine different joint speeds and grasp states of the robot hand. For extracting generic behaviors, the system finds segments with the same initial-end situation. These segments are grouped and combined into a single representative behavior primitive, computed by taking the average of initial and final velocities. As in previously mentioned papers, this work only considers single primitives, and sequential combination of motor primitives is not studied.

II. CONTINUOUS GOAL-DIRECTED ACTIONS

Continuous Goal-Directed Actions (CGDA) is an imitation framework where actions are analyzed in terms of their effects on objects. Objects features are recorded during human demonstrations. These features form an n-dimensional feature space, where n equals the number of tracked object features. Tracked object features not only are spatial positions; examples of tracked features can also be color, area, or weight among others. Each human demonstration is represented as a feature trajectory in this feature space. Actions are generalized by analyzing these feature trajectories.

The CGDA framework was introduced in [4]. The focus on the current work is on execution. However, we outline the generalization explained in [4] for a better understanding of the rest of the paper.

Human demonstrations are represented by a sequence of discrete points in the feature space. The set of demonstrated action repetitions leads to a point cloud in the feature space. A representative feature trajectory is extracted from the cloud. This feature trajectory represents changes produced in the object features when an action is performed on it. Generalization is composed by the following three steps.

1) Time Rescaling: Before inserting an action repetition in the point cloud, each repetition must be normalized in time. All normalized feature 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 N temporal intervals. The number of intervals is computed from the average duration of the original repetitions by fixing one interval per second. The representative point p of each interval is extracted as $p = \frac{1}{p_{int}} \sum_{i=0}^{p_{int}} X_i$, where p_{int} is the number of points in the interval and X_i represent the vector of features for a point. The result is a vector of interval average points.
3) Linear Interpolation: Once we have each interval average point, we have to connect them to create a generalized feature trajectory of the action. As an interpolator, we use a linear Radial Basis Function (RBF) which returns a generalized feature trajectory. Notice that this generalized feature trajectory represents how the object's features are changing through time.

With these steps we obtain a generalized feature trajectory from a set of repetitions of a demonstrated action. This generalized feature trajectory represents the changes produced in the object, and not the kinematic or spatial parameters of the action. When a feature trajectory is represented in a plot, each axis represents a different object feature. Feature trajectories can also be plotted against time as an axis.

As kinematic parameters are not explicitly provided, tasks encoded as CGDA require a particular technique to be reproduced. In previous works [4], the tested algorithm was based on Evolutionary Computation. We now aim to improve this strategy by using guided motor primitives.

III. GUIDED MOTOR PRIMITIVES

Our work on CGDA execution can be seen as a search of inverse models (compute the action policies that can generate a given effect) based on task goals. We benefit from human-robot interaction to create a library of Guided Motor Primitives (GMP). These primitives are created by extracting segments of movements from a random human guidance with the robot. Our aim is to make the robot capable of discovering motor primitives and being able to perform tasks with them, similar to motor babbling. The process starts with the robot arm being moved randomly by the human (Fig. 1).

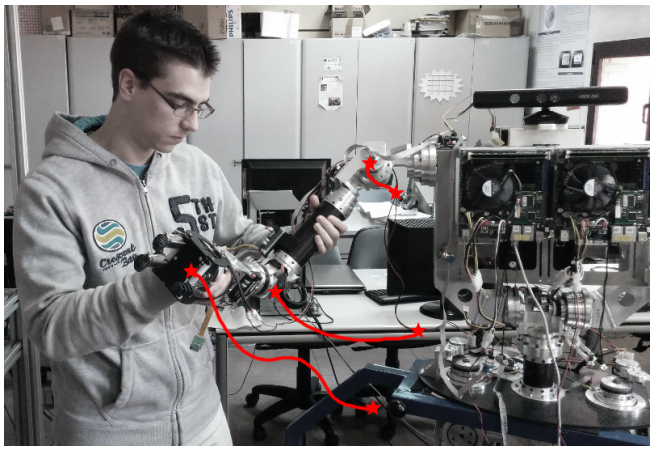

Fig. 1. Human guiding the robot in a random spatial trajectory. The red lines represent the spatial trajectories the joints follow during the guiding.

This guided exploration generates a joint trajectory in the joint space (an example using three joints is shown in Fig. 2). The joint trajectory is split in small segments of τ seconds and all the segments are stored in a set.

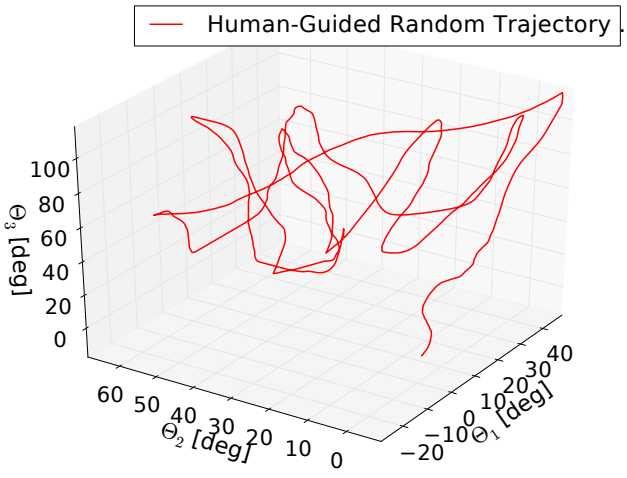

Fig. 2. This random joint trajectory involves 3 joints of the robot arm. It is generated when the human performs some random movements with the robot arm.

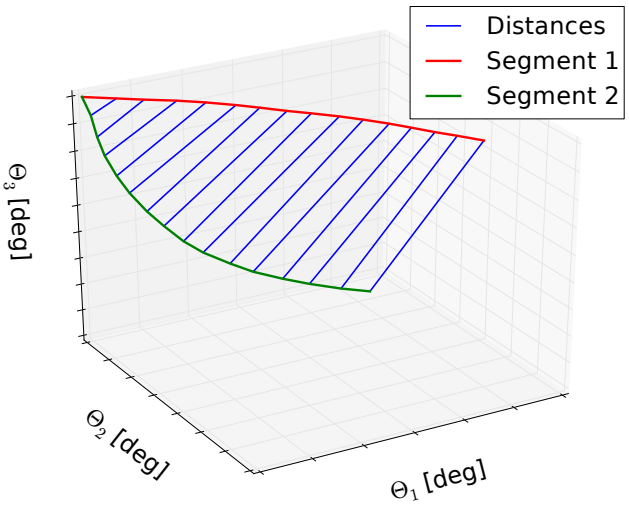

Fig. 3. An example of comparison between two random segments extracted from the previously shown joint trajectory. Both segments have a duration of $\tau = 0.5$ s and the blue lines represent the distances between pairs of points.

The process to select the primitives from the set, starts with the first segment, which is considered the first GMP. Then, the second segment is compared with the first. If it is different enough, it is considered a new GMP, if it is not different enough, we discard it and continue evaluating the next segment. Once more than one primitive is stored, the next segments are compared with all the stored primitives.

For comparison, the primitives are transformed to be relative to the origin of coordinates, while conserving their shape. The comparison between two discretized segments is performed by applying the Euclidean distance over all pairs of points (one belonging to the primitive and the other one to the segment being evaluated) aligned in time. This analysis returns a single value J of discrepancy which is then used to store or reject the segment. If J is lower than a hand-crafted threshold ξ, the segment is discarded. An example of comparison between two random segments is shown in Fig. 3.

After 47 seconds of random human guidance, 94 segments were generated with $\tau = 0.5$ s, and 47 segments when selecting $\tau = 1$ s. In function of ξ (the threshold of similarity), these segments lead to sets of GMP with different number of primitives. Examples of GMP can be found in Fig. 4.

One advantage of this library of GMP is its re-usability. The primitives are task-independent and can be applied to different situations. This affirmation will be demonstrated in the experimental section, where two experiments are performed with the same set of segments. Another advantage is that, in case a task cannot be accomplished with the current library of GMP, two mechanisms exist to increase its number. First, it is possible to perform more human guided random interactions to obtain more primitives e.g. if new joints are available, a new human-guided interaction can reach currently unknown areas of the joint space leading to new primitives. A second mechanism is implicit in the threshold of similarity ξ. As it is a manually set parameter, its value may be decreased, incrementing the number of GMP.

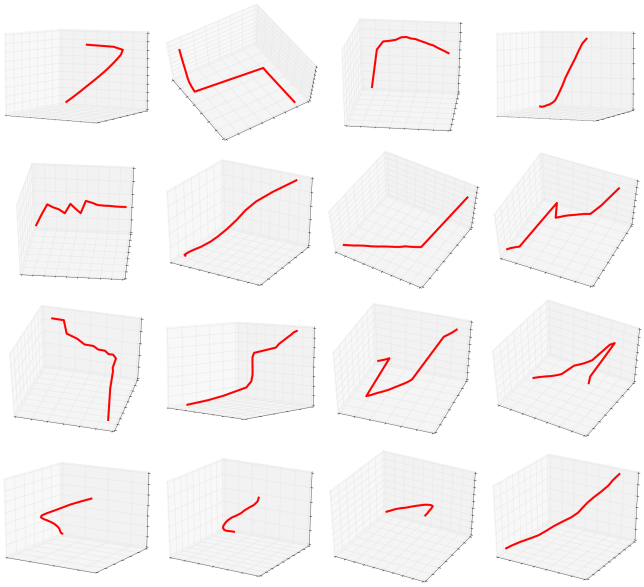

Fig. 4. Several examples of guided motor primitives in the joint space. They were extracted from the joint trajectory shown in Fig. 2.

A. Sequential Incremental Combinatorial Search

Once the GMP library has been created, we must focus on how to combine sequentially the primitives for reproducing tasks encoded as CGDA. The encoded features usually have a temporal dependency, where each step in the feature trajectory is dependent of the previous feature point. For instance, painting a surface a depends on how much of the surface has already been painted. For these kind of features, executing the primitives that will achieve each point of the feature trajectory independently is not feasible.

For this reason, a search method called Sequential Incremental Combinatorial Search (SICS) has been developed. SICS is essentially a tree search algorithm working in a

breadth-first manner within trees with incremental depths. In our case, each node represents a GMP, and each edge has an associated value which represents the cost of traversing a path from the initial node to the current one. Traversing a path is the result of executing the sequence of primitives (nodes) of the path. The path cost is calculated in the feature space and it is the difference between the feature trajectory generated when the primitives of the path are executed in order, and the provided generalized feature trajectory. A summary of SICS can be found in Algorithm 1.

Two elements make SICS different from a standard breadth-first algorithm. First, the level of depths of the tree are incremented if no solution is found (a standard algorithm would stop here). The number of maximum levels is determined by the user, allowing even an infinite number (completeness is not guaranteed in this case). Second, as the primitives are contained in the joint space but the evaluation is performed in the feature space, the cost of sequentially executing two primitives can be lower than executing a single one. This fact prevents from discarding branches of the tree and forces an exhaustive evaluation through all nodes.

Let us outline the working mechanism of the algorithm. Assuming we provide a generalized feature trajectory as reference, and discretizing this trajectory in points, SICS sequentially searches for a solution for each feature point. A solution is considered valid when the generated joint trajectory produces objects features that are similar to those of the reference. SICS first evaluates the costs of nodes belonging to the first level of depth of the tree of primitives. If the cost of the path to reach a given node is lower than a manually set parameter ε, it stops searching, stores the path, and continues searching a solution for the next point of the feature trajectory. If the cost is not lower than ε for any node of the level of depth, the tree is expanded one level of depth, and the costs of sequentially executing the two depth-level paths are analyzed (Fig. 5).

The same process of search and expansion is iteratively followed until a solution is found for all feature points (the error is lower than ε for all points), or the tree has expanded d times without finding an acceptable solution.

The path cost is evaluated in the feature space by calculating the Euclidean distance between the discretized feature trajectory that results from sequentially combining the primitives, and the discretized generalized feature trajectory.

IV. EXPERIMENTS

Two experiments have been performed. Both experiments are performed in a simulated environment with a model of the humanoid robot Teo [18], using the previously discovered human-guided primitives. The first experiment involves only object spatial features, and its objective is to generate a robot joint trajectory which leads to a feature trajectory equal to the objective one. This feature trajectory is similar to a cleaning movement, and it is encoded as a CGDA where the object tracked is in one of the robot's hands. The second experiment consists in having the robot paint a wall. The painting process

Algorithm 1 Sequential Incremental Combinatorial Search (SICS)

```
1:  procedure SICS(X, ε, d) ▷ X is the feature trajectory. ε
    is an error parameter. d is the maximum allowed depth.
2:      P ←()          ▷ P is the sequence of primitives used.
3:      for i < numberOfPoints(X) do
4:          P ←search(P, ε, d)
5:      end for
6:      return P
7:  end procedure
8:  function SEARCH(P, ε, d)
9:      depth ← 1
10:     while depth < d do
11:         N ←breadthNodesIndexes(depth)
12:         for j < length(N) do
13:             C ←pathTo(N_j)
14:             f ←execute(P + C)
15:             cost ←compare(f, X_i)
16:             if cost < ε then
17:                 P ←add(N_j)
18:                 return P
19:             end if
20:         end for
21:         expandTree(N)
22:         depth ← depth + 1
23:     end while
24:     P ←addLowestCostPath()
25:     return P
26: end function
```

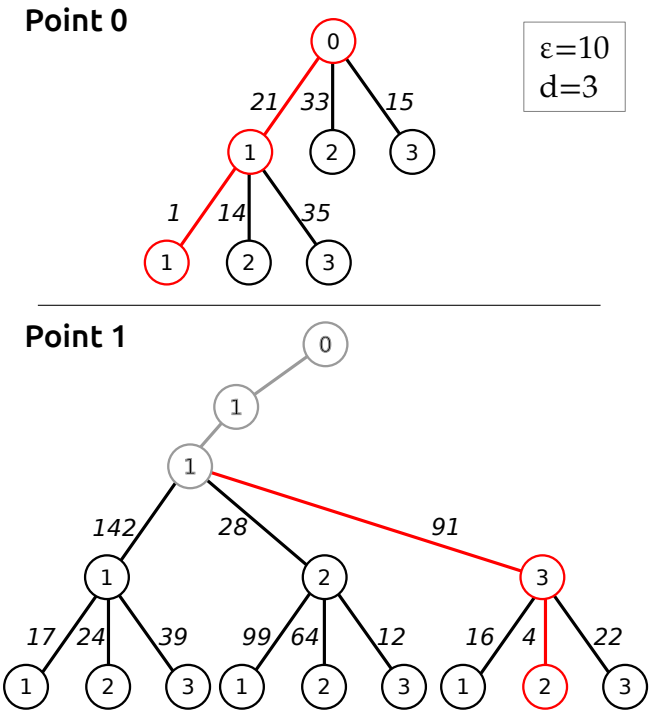

Fig. 5. SICS example for a two-point feature trajectory. For the first point, the nodes are expanded and a breadth-first search is performed until the cost of one of the nodes is lower than a threshold ε. If this occurs, the path to the node is stored and the process continues for the next point. If not, the tree is further expanded and the search repeats.

is encoded as a CGDA, but this time the object tracked is the wall.

A. Spatial Task: Cleaning

The goal is to reproduce a generalized feature trajectory extracted from a set of human action demonstration repetitions. This generalized feature trajectory contains only spatial features (X,Y,Z). The demonstration repetitions which leaded to the generalized action were recorded using a real Kinect device tracking a colored marker. Seven repetitions of the demonstrated action were recorded and used to generate the generalized action. The action can be described as: keeping object orientation fixed, move it over the perimeter of a circle of 30 cm of diameter for one revolution. For simplicity of explanation, let us name this action as 'clean'.

For this experiment several values of ξ were used to generate different sets of primitives. With these primitives, SICS was applied to find a primitive combination which obtains an error lower than $\varepsilon = 20$ for each point of the generalized feature trajectory. This ε represents a 20 mm error in this experiment because the feature space was set to millimeters. The resulting feature trajectories for the cleaning task can be seen in Fig. 6.

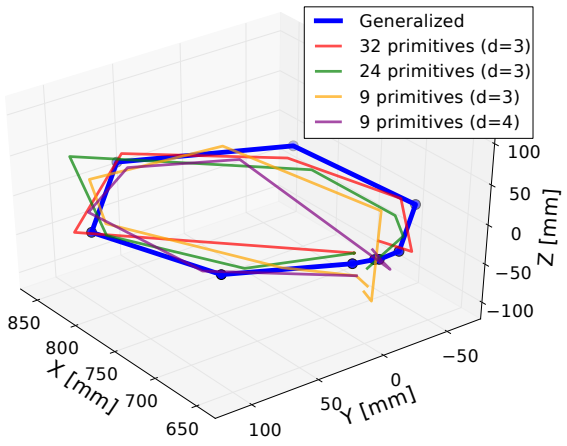

Fig. 6. Cleaning: Plot superposing the generalized feature trajectory (blue) and several feature trajectories resulting from executing the sequence of primitives selected by SICS. Results are dependent on the number of primitives available to choose from (given by the threshold of similarity ξ) and the maximum tree depth d. The primitives used belong to the same set of segments, but in function of ξ, the most diverse are selected. Notice the plot is not representing joint or spatial trajectories, but feature ones.

In general, allowing a greater maximum tree depth d increments the possibilities of finding a solution, but results in an exponential evaluation complexity. The use of a larger set of primitives leads to a greater number of possible combinations, so the algorithm is more likely to find a solution.

B. Non-Spatial Task: Painting

In the second experiment the task goal is to paint a wall, without teaching the robot the necessary motor parameters. In the simulated scenario, there is a wall, composed by 16 small gray squares, in front of the robot. The action goal is to change the color of all the squares. The generalized action has been created synthetically and it has only one feature (plus time). This feature represents the percentage of wall that has been painted, which increases constantly with time. The generalized feature trajectory is split in 16 points.

The generalized feature trajectory is used as the reference and SICS is used to find primitive sequences. Every time the robot hand gets closer than 120 mm to a wall square, this square changes its color. This is a simplification to emulate the real action of sliding a paintbrush over a surface. Cost is evaluated when a primitive sequence is executed, by analyzing wall features (by counting the number of colored squares).

Several values of ξ were used to generate different sets of primitives. With these primitives, SICS was applied to find a primitive combination with an error lower than $\varepsilon = 0.1$ for each point. This ε represents a 0.1% error in this experiment as the feature space was set to represent the percentage of the painted wall. The resulting feature trajectories for the painting task can be seen in Fig. 7.

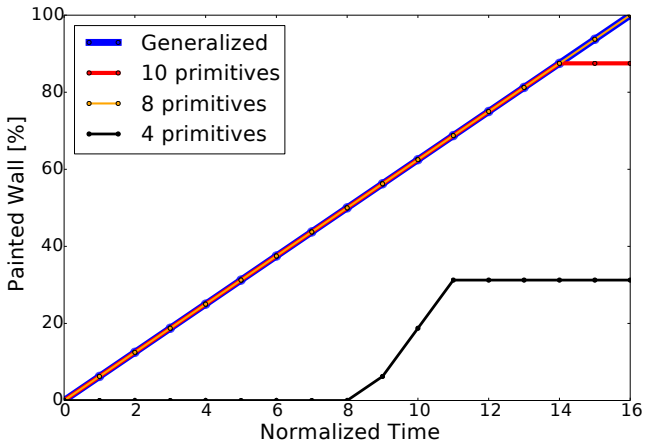

Fig. 7. Painting: Plot superposing the generalized feature trajectory (blue) and several feature trajectories resulting from executing the sequence of primitives selected by SICS. Results are dependent on the number of primitives available to choose from (given by the threshold ξ). Remember the plot is not representing joint or spatial trajectories, but feature ones.

The results show that the system can accomplish the painting task with a small number of primitives (example in Fig. 8). By increasing the number of primitives, the probabilities of a correct performance rise. However, a new primitive may alter a path and lead to worse results for posterior points. This is the case of the figure when using 10 primitives.

V. DISCUSSION

An improvement over our previous work [4] is the reproduction of non-spatial feature tasks, such as those involving color. In [4] we were restricted to spatial characteristics, as we had not found a method with acceptable results. The use of guided motor primitives is new in the framework and the inclusion of a human in their generation is also valuable. By benefiting from human-robot interaction, the

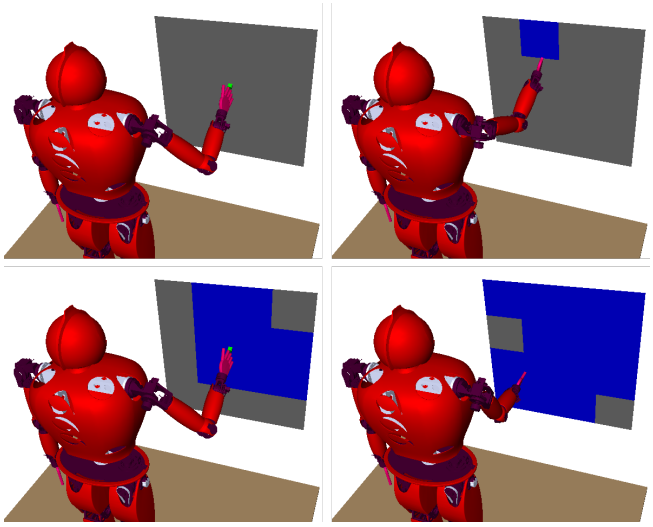

Fig. 8. Several screenshots taken during the painting experiment. Motor sequence obtained from combining primitives using SICS.

complexity of the framework is reduced when generating primitives. Other self-discovery approaches, such as motor babbling, requires programming joint-limits and computing self-collisions. Despite GMP and motor babbling aim to fill similar technological niches, our approach simplifies the process as it is the human who leads the exploration of the joint space.

VI. CONCLUSIONS

This paper describes the discovery and combination of Guided Motor Primitives (GMP) to perform goal-directed actions. The primitives are extracted from a random joint trajectory generated when interacting with a human. After only 47 seconds of human interaction, the robot has learned enough primitives to perform several tasks in simulation. By combining the discovered primitives with the Sequential Incremental Combinatorial Search the robot is able to perform the execution of tasks encoded as Continuous Goal-Directed Actions (CGDA).

Regarding the experimental results, we have demonstrated that CGDA encoding is feasible for action reproduction even with non-spatial features and also that GMP are reusable for different tasks. The results show an increase in performance when using a greater number of primitives, as it has more combinations to choose from. Furthermore, longer combinations of primitives also tends to improve the results.

Future works will involve adapting the proposed GMP to work on real robotic platforms. This adaptation will imply smoothing final joint trajectories and also implementing a learning process during the tree search, as it is not feasible to evaluate all primitives combination for each task.

ACKNOWLEDGMENT

This work was supported by ARCADIA (DPI2010-21047-C02-01), funded by CICYT, and RoboCity2030-II-CM (S2009/DPI-1559), funded by Programas de Actividades I+D in Comunidad de Madrid and by EU.

REFERENCES

[1] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 2, pp. 44–54, 2010.

[2] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[3] C. L. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Interdisciplinary Approaches to Robot Learning*, vol. 24. World Scientific, 1999, p. 136.

[4] S. Morante, J. G. Victores, A. Jardón, and C. Balaguer, "Action Effect Generalization, Recognition and Execution through Continuous Goal-Directed Actions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] Y. Mohammad and T. Nishida, "Tackling the correspondence problem," in *Active Media Technology*. Springer, 2013, pp. 84–95.

[6] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 299–304.

[7] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. Van Schie, and H. Bekkering, "Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning," *Robotics and autonomous systems*, vol. 54, no. 5, pp. 353–360, 2006.

[8] R. Saegusa, G. Metta, G. Sandini, and L. Natale, "Developmental action perception for manipulative interaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4967–4972.

[9] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[10] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2002, pp. 1398–1403.

[11] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[12] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann, "Towards robot skill learning: From simple skills to table tennis," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 627–631.

[13] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[14] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[15] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008, pp. 834–839.

[16] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[17] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 3260–3267.

[18] S. Martínez, C. A. Monje, A. Jardón, P. Pierro, C. Balaguer, and D. Muñoz, "Teo: Full-size humanoid robot design powered by a fuel cell system," *Cybernetics and Systems*, vol. 43, no. 3, pp. 163–180, 2012.