

Compressed Sparse Code Hierarchical SOM on Learning and Reproducing Gestures in Humanoid Robots

Georgios Pierris and Torbjørn S. Dahl

Abstract—Compressed Sparse Code Hierarchical Self-Organizing Map algorithm (CoSCo-HSOM) is an extension of ideas existing in the gesture classification and recognition research area. Using as foundation Hierarchical Self-Organizing systems and cognitive models introduced by neuropsychologists, we propose the CoSCo-HSOM algorithm introducing novel characteristics. During the training phase of the algorithm, activity lists are used on a layer level, instead of decreasing activity levels in each node. Furthermore, we expand this approach to eventually reproduce a generalized learned task on the Aldebaran Nao humanoid robot, by using only information of the current robot’s posture. Finally, a comparative analysis with the Gaussian Mixture Model approach on the same experiment using the same training data, supports the effectiveness of CoSCo-HSOM.

I. INTRODUCTION

Nature has always been giving answers to our problems, but it seems that researchers has been selective on which models to use. Biologically inspired robots [1] bypass the slow process of evolution by replicating animal or human embodiments for free. On the other hand, even though Cognitive Science has not reached the point yet to reverse engineer human brains, recent studies hypothesize in great detail the learning processes infants follow [2]. In this work, we model robots as infants with no prior experience, assuming both organisms utilize “blank-memory” or “no-experience” systems. Infant cognition can be modeled as a hierarchical bottom-up approach and the usage of neural networks as layers in the hierarchy is a natural selection [3].

In the context of this work, we propose a Compressed Sparse Code Hierarchical Self-Organizing structure (CoSCo-HSOM) for learning and reproducing gestures on humanoid robots. The structure we used is based on a Hierarchical Self-Organizing Map (HSOM) [4], but we introduce a novel approach in compressing the sparse code signals. These signals are used as inputs in layers higher than the bottom. A common approach is to create in every layer, large patterns of the last active nodes. By active we mean a match of the input with a particular node during the training phase. Depending on the memory size of our structure, the pattern “remembers” the last n nodes and sets the *activity level* to zero in all

The research leading to these results has received funding from the European Commission’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 231500.

G. Pierris is with the Robotic Intelligence Lab, University of Wales, Newport, Allt-yr-yn Campus, Newport NP20 5DA, United Kingdom. georgios.pierris@newport.ac.uk

T. Dahl is a Reader at the University of Wales, Newport Business School and head of the Robotic Intelligence Lab, University of Wales, Newport, Allt-yr-yn Campus, Newport NP20 5DA, United Kingdom torbjorn.dahl@newport.ac.uk

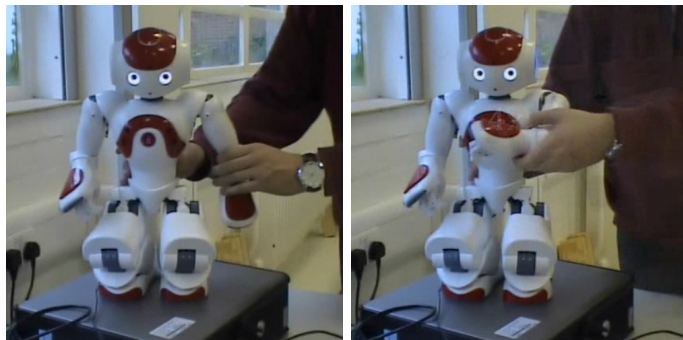


Fig. 1. Aldebaran Nao’s initial/final posture (left) and “reading watch” posture (right).

the other nodes. For the active nodes, a different value is assigned in every step and depends on a decreasing factor. A naive model is to linearly decrease the value of the node, i.e. in a system keeping track of the last four nodes, the activity levels can be 1.0, 0.75, 0.5, 0.25, and 0.0 to older or inactive nodes, with 1.0 being the most recently activated node in a layer. The values themselves are not important in the structure, it is crucial though to follow a decreasing pattern, because this is the information we use to learn sequences of primitives. However, in this previous work, there is a large overhead in terms of computational time, since in every training step, we have to update that signal by running through each node in all corresponding layers. The complexity might be linear, but doing it in every training step makes it time consuming. Instead of using activity levels in every node, we indirectly monitor the activity in every layer, rather than in a single-node perspective, by updating a fixed-size FIFO queue for every layer keeping track of the recently active node’s IDs in every layer. Each queue comprises a signal, which can be fed up in higher layers as input, instead of sending a longer version of the same information. The complexity of updating the queue is constant, as we only push and pop the best and the oldest matching unit respectively in every step, without the need of running through every single node.

The algorithm supports through the use of a single memory structure the integration of both short term memory (STM) and long term memory (LTM). The weights of the HSOM constitute the LTM. During the learning phase, the activity lists are used as inputs to nodes on higher levels in the hierarchy and influence how the weights are adjusted, producing a transferral of information from STM to LTM. As a result, different sets of node IDs come to encode different observed

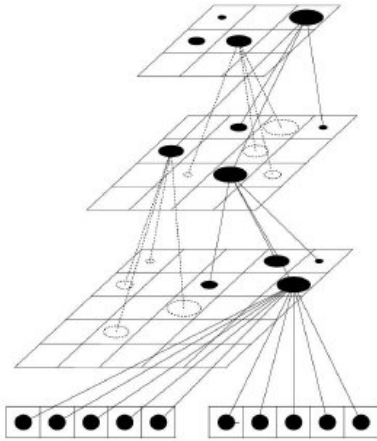


Fig. 2. Short term memory as activation in a hierarchical SOM.

sequences. A single node represents a finite sequence of observations and, as such, may be reused several times in a longer sequence or by different sequences. This reuse will be encoded by different non-zero weights in the map, where the node has the role of an input, i.e., the map on the layer above in the hierarchical structure. Figure 2 presents a CoSCo-HSOM structure illustrating activity levels as ovals of different size. In CoSCo-HSOM, each node represents a principal component of the observed sequences of states and actions. Figure 2 also illustrates how the most recent actions are recorded in low level STM, while sequences of more abstract actions are represented on higher levels. The dashed ovals indicate sequences of a loose state-action model, i.e., nodes, that are referred to by nodes that are present in STM, but are not themselves present, i.e., the abstract sequences containing these nodes are currently active, but the nodes themselves no longer have a raised activity level. A more thorough description of the CoSCo-HSOM and its distinctive features will be discussed in Section II-C.

II. EXPERIMENTAL SET-UP

A. System Architecture

Hard-coded robotic behaviors introduce many problems to researchers, such as the lack of generalization and the absence of re-usability. In order to compensate these problems, focus has been given to Imitation Learning, Programming by Demonstration and other machine learning solutions [5]. In the context of this work, we are mostly interested in the gesture production, and particularly in generalizing a motion by learning gesture patterns. In the proposed approach, we develop a novel Hierarchy of Self-Organizing maps to reproduce gestures in humanoid robots, by learning timed sequences of robotic postures. Therefore, a simplified version of robotic motion, is the execution of these robot posture sequences. Posture execution is a control-based transition (linear, smooth etc.) from one pose to the other in the configuration space. As soon as we focus on

a single posture, the latter can be fully described by a time independent sequence of joint values. From the joints' perspective, motions can also be represented as parallel simultaneous execution of all robot joints. From that point of view, the timed sequence of values on each joint, are represented as a time-series. In our architecture, we propose an open-loop system that allows off-line training of a robot through supervised demonstration sessions. During these sessions, we record posture data and feed them afterwards to train the hierarchy. Once the hierarchy have converged to its final weights in every layer, we can then save it in binary or plain text format taking the advantage of the minimal memory requirements. The memory needed is fixed no matter the length of the gestures and minimal compared to large dictionaries generalized versions of gestures. Finally, by loading the structure to the robot, we are able by using only the initial posture as input, to reproduce a complete gestures.

B. Hardware

Nao [6] is a relatively small humanoid robot. It has been developed by Aldebaran Robotics based in Paris, France and is a 58 cm tall robot weighing 4.3 Kg, utilizing 25 degrees of freedom (Academics Version). The commercial release of Nao has been planned in the near future, however Aldebaran has achieved to promote Nao as an educational robotic platform and a family entertainment robot affordable to most budgets. Even though Nao's capabilities cannot be compared to those of other humanoid platforms, it can potentially be considered as a benchmark platform, due to its large popularity especially among European Institutes and RoboCup competitions. The Nao programming environment is based on the proprietary Naoqi framework, which serves as a middle-ware between the robot and high-level languages, such as C, C++, and Python [7]. Serving the generality of our approach, this environment is being used only in terms of strictly sending commands to the Nao. In that sense, the architecture is platform, and application independent. Finally, there are realistic computer models of the Nao robot available for the Webots[8] robot simulator and others. It is important to execute any reproduced motion, before trying any gesture on the real platform, as faulty controllers can be produced by machine learning algorithms.

The training data are captured from the encoders of each joint. Specifically, during the training phase the "expert" user manipulates the robot's joints in the desired direction and desired speed. The joints are either being set in passive mode to safely move them, or dynamically allow the guidance by the demonstrator through touch interpretation [9]. A simple logger, records the data in high frequency (40Hz) by reading the sensory information received by the motors, saving at the same time all the data for further processing. It is a good practice to allow high data resolution by recording motions in high frequency, but in practice our approach has been proven effective even when using way too low recording frequency. Furthermore, by taking advantage of the nature of CoSCo-HSOM algorithm, there is no restriction on what training

data are being used, in terms of their semantic meaning such as captured by motor encoders, motion capture devices, or kinematic analysis, as long as there is a controller to eventually inverse the learned data to a motion.

C. Compressed Sparse Code HSOM

Research has been proven vast on the generalization of complex robotic gestures [10], [11], [12]. From our perspective, we hypothesize that biological models consist a comprehensive approach to the problem, thus we rely on neural networks. In this work, following the constructive model of cognition on infants [2], we have built an hierarchy of Kohonen Self-Organizing maps [13]. Every layer has a different set-up from the others, in terms of the learning factor, learning radius, dimensions etc. and we randomly initialize the weights in every map. The bottom layer is used for learning different body postures with a slight variation of separating the state-action input in two different inputs. The state is the complete posture or a part of it, whereas the action is considered to be the state in the next timestep, which is apparently the next posture. For example, at timestep t the input will be

$$S_t = \{\cup x_t^i \forall i\}$$

, where i is the union of the robot joints (or other type of variables depending on the context of the problem we want to solve). The other input will be the

$$S_{t+1} = \{\cup x_{t+1}^i \forall i\}$$

The action will then be the linear transition

$$S_t \rightarrow S_{t+1}$$

in the configuration space. To this point, in the bottom layer we are only able to do one step prediction, of matching input state with a node and predicting the next state. This transition can be loosely expressed as a gesture primitive. If we follow a path from the top to the lower levels, we retrieve complex gestures expressed as a combination of primitives.

1) *Training Phase*: Having collected the training data, the CoSCo-HSOM algorithm has to be trained in a sequential way, in contrast to usual random feeding of Kohonen maps. The order of the data used is crucial, as it implies the steps you have to follow to accomplish the task. The algorithm does not learn just a state-action input, but complete motions through a series of state-action pairs. Another distinctive characteristic, is that in every layer, we keep updating a fixed size FIFO activity list with all the IDs of the nodes. In our experiments, the ID is the (x, y) pair in the discrete 2D grid of every map. By finding the best matching unit in every training step in the bottom layer, we insert the ID of that node, and we pop the oldest node in the activity list, if the latter is full. The size of the list is set by the user and we call it *memory*. An important step is to clear all the activity lists every time a complete motion has been fed to the algorithm. Otherwise all the training session is considered a long gesture causing, as in general this approach is sensitive to replicating the same motion at the reproduction phase.

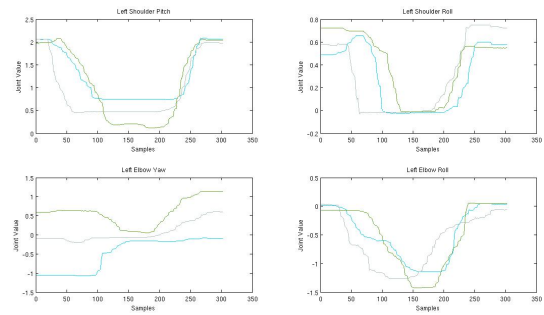


Fig. 3. bla bla bla bla adflkjfd learned

In order to compress long gestures, every layer has to be trained exponentially as we go in higher levels based on the update period we decide. In every training step, the bottom layer is being trained, but the upper layers are trained every $updatePeriod^{level}$ steps, where $updatePeriod$ is again hard-coded by the user, and $level$ is the level of each layer counting from zero (the bottom layer). For example, if the $updatePeriod$ is equal to two in a three layer architecture, after training step 7, the bottom will have been trained 7 times (at steps 1, 2, 3...), the second layer 3 times (at 2, 4, 6) and the third only 1 (at 4). Eventually, even long sequences can be encoded in a small architecture without losing valuable information of the motion. Additionally, instead of using a complete sparse signals of activity map in every layer, we suggest using a *memory* bounded number of the last occurrences. In that way, we compress even more the memory requirements and running cycles of the hierarchy.

2) *Reproduction Phase*: The gesture reproduction by itself is novel in Hierarchical Self-Organizing maps. After the training phase, related approaches focus on the gesture recognition[14], rather than the reproduction in robotic systems. In this work, we propose a novel approach of reproducing a learned gesture for robots. According to the infant cognition model [2], infants build complete behaviors in hierarchies, and when it is time to execute a gesture or a behavior in general, they start from the bottom layer going higher, until the brain gets confused. The source of the confusion is not clear yet, but we assume it is due to the addition of irrelevant noise or similar problems. The higher we go the more information we gather. As soon as, the brain cannot go in a higher layer, we fall back and from that point reproduce the gesture going towards the bottom layer executing the motion primitives. In order to start reproducing though, the controller needs some short of history to start predicting next motions. The history is a small number of matches in the bottom layer, that allows us to go higher in the hierarchy and get more information concerning the gesture.

Considering that as a drawback, our goal is with no prior knowledge of anything, and only using the trained HSOM to be able reproduce a complete gesture. The only information we need is the current state of the robot. CoSCo-HSOM starts a first prediction of the next pose by using the initial posture

as input and getting the output at the bottom layer. We follow the same process, until we have built a “considerable” amount of matches at the bottom layer. “Considerable” is expressed as an *offset* of the maximum *memory* defined for the system. For example, if the *memory* is four, then a node in the second layer represents four nodes in the bottom. If we have 3 matches in the bottom layer, we can confidently predict the fourth by going one layer up and finding the best matching unit for the three occurrences ignoring the fourth. As soon as, we predict the fourth node, we execute it and continue in the same way to the upper layers. The *offset* is customizable and allows us to study the effect of larger offset, making aggressive predictions. One could argue that the first steps of the reproduction are just small naive gesture primitives and make the reproduced gesture poor. That is true, but we expect to get better predictions in the short future, where we will have built the required history to find occurrences in higher levels. The higher we go, the more confident we are about the prediction we do. Finally, as soon as we decide to move downwards the predicted node, points to *memory* different, or the same nodes in the directly lower layer and again each node to other nodes lower. Predicting a sequence of primitives is a tree traverse problem. Particularly, we follow a Depth First Search approach starting from the older node. The older node is in the end of every activation list (we insert from the beginning and pop from the back), and thus the oldest primitive can be found by following the rightmost node in every activation list, and the newest the leftmost. Then we simply append the pattern of postures to the so far predicted gesture. The final step is to eventually execute the transition from posture to posture.

One might argue that, because of the nature of the Kohonen maps all nodes have been initialized with random weights. This will result in only having floating point number in every node. In the bottom layer this is not a problem, as the posture of the robot has the same structure, but when it comes to learning the activity lists in higher layers, then the IDs will be natural numbers such as the indexes in 2D matrix, or the position in the topology. In order to solve that practical problem, you can either use the rounded signal (might need the floor of the ID, if it didn’t converge fast) to get the indexes, or find the closest node by calculating the euclidean distance of the prediction and the map in the lower layer, but this solution introduces many CPU cycles, losing the advantage of predicting gestures with linear complexity.

D. Configuration

Our system is fully customizable, by loading all the information from an XML file. *memory*, *updatePeriod*, *state-action*’s size, and the *trainingSteps* can be changed on demand. Additionally, the height of the hierarchy, the size and the topology of every layer, the learning radius, and the learning factor are customizable as well, allowing to the user experiment with various setups to match the problem’s needs. It worths mentioning, that the learning factor has to be different in every layer, allowing to a small learning factor at the bottom and more aggressive factors as you go higher. It is

obvious that the higher layers are being trained infrequently. With our approach we eliminate the trade-off of finding a good learning factor for all layers, which is not possible especially in our work, where we tend to train hierarchies, even to the height of six or more.

III. EXPERIMENT

We present an experiment with the Aldebaran Nao robot learning a simple gesture. We eventually compare our motion with the Programming by Demonstration approach, which successfully uses Gaussian Mixture Models [15]. We also argue that our method is generic and transferable to other types of robots as well, since the CoSCo-HSOM algorithm is model free. By blindly training the HSOM algorithm for different robots, the acquired gesture will directly refer to the specific platform. In order to verify the effectiveness of the reproduction phase, we learn a simple gesture similar to raising the left arm in the position similar to “reading a watch”. The initial and final positions are the same to add more complexity to the problem.

The task consists of training the Nao robot, with the help of an expert supervisor. It learns and successfully reproduces the desired gesture. In Figure 1 you can see the initial position, which is the same with the final one, and the key pose of “reading the watch”. In this experiment, we have used 3 training sessions of the similar gestures. It is important to make clear, that for this experiment we have only used the left arm’s four DOFs as a training set, and not the complete robot’s configuration. The time-series of all three training sessions for each joint are shown in Figure 4. Obviously, the training data have not been processed in any way, and this is one of the advantages of this approach. CoSCo-HSOM does not need any pre-processing steps, and raw data can be used without problems. In Table III you can see the exact configuration used in our experiment (all layers have Orthogonal topology, Hexagonal is also an option for all). The reproduced gesture of the trained hierarchy is the black line in Figure 4.

Hierarchy	Learning Factors / Grid Size
Height : 6	Level 5 : 0.8 / 1 x 1
Memory : 4	Level 4 : 0.6 / 2 x 2
Update Period : 2	Level 3 : 0.5 / 6 x 6
Training Steps : 5000	Level 2 : 0.3 / 8 x 8
State Length : 6	Level 1 : 0.2 / 10 x 10
Action Length : 6	Level 0 : 0.1 / 16 x 16

A. Future Experiments

The presented experiment is only a part of our initial results and serves the purpose of a qualitative comparison of the reproduced gestures against other methods. From this point, we are tempted to explore the effect of different configurations in various gestures and more importantly how the CoSCo-HSOM can handle multiple gesture production from a single hierarchy. We expect to encounter some difficulties in the reproduction phase, where having learned two or more different gestures, which share similar, if not the same initial postures, will definitely create perturbations in the motion.

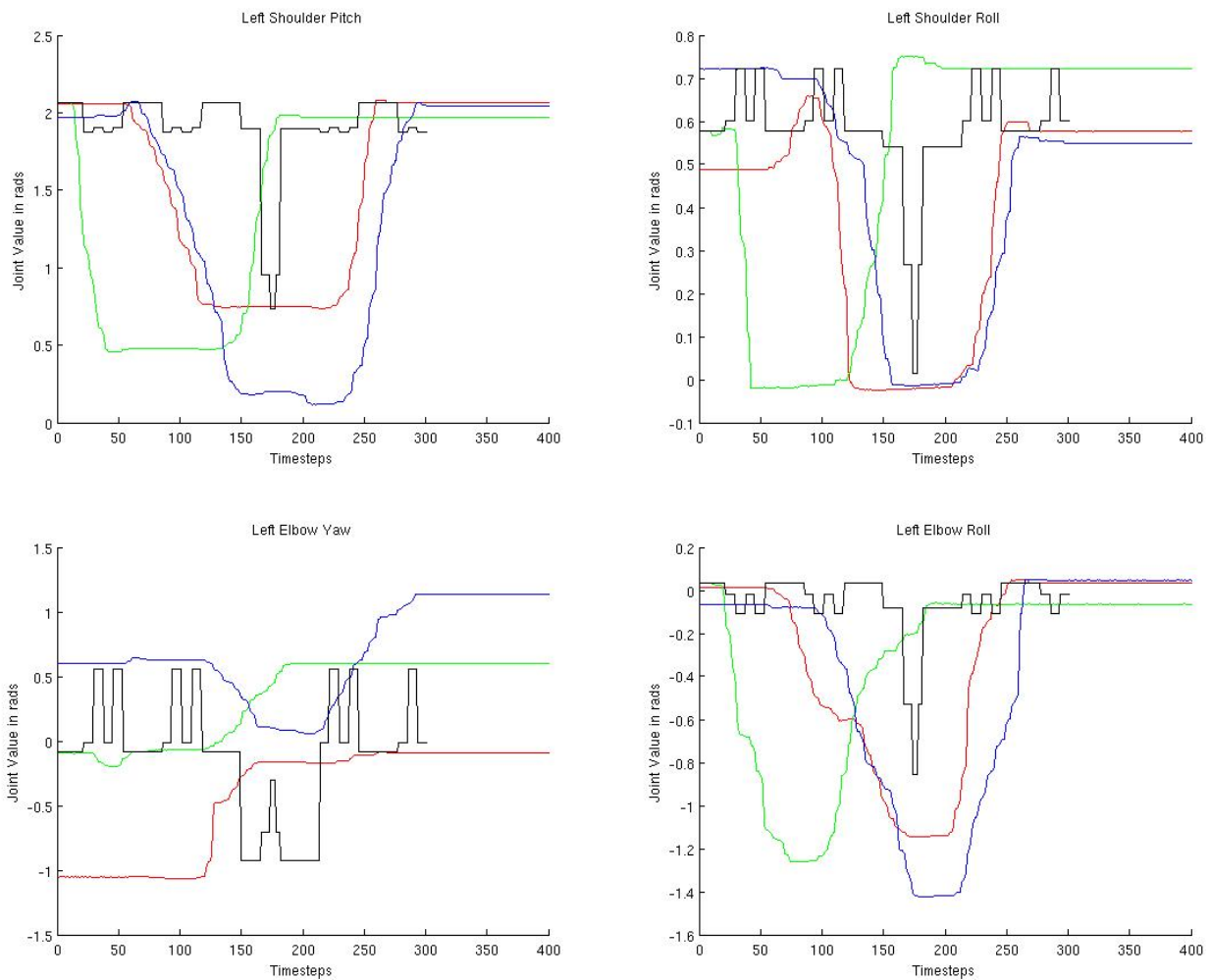


Fig. 4. Raw training data from three training sessions.

Taking advantage of the modular architecture, we are now researching methods of building hierarchies of trained HSOMs. This work can potentially be used in learning behaviors using a sequence of complete gesture primitives. Our future plans include an extensive analysis of the limits of this work and other domains, where the CoSCo-HSOM could be potentially proven useful. We are also confident that sensorimotor control applications will take advantage of this approach. This area is open now and we are looking forward to fitting our work in the reproduction of cognitive models in robotics.

IV. DISCUSSION

The cost of humanoid robots has been dropped significantly in the last years. Consequently, all the more research groups, and even independent researchers are being attracted by these robots. Therefore, many approaches similar to gesture learning have been proposed in the literature. An important work is the use of Gaussian Mixture Models [15], [16]. As our work is novel in the reproduction of the gestures, we compare our solution with the GMM model. Even though the algorithms have almost nothing common share, they

both solve the same problem. In order to be as objective as possible, we decided to use the same training data in both approaches. The raw data of the training sessions are in Figure 4 (excluding the black line).

One of the key features of CoSCo-HSOM algorithm is ability to directly feed the algorithm with training data, whereas in the GMM model a pre-processing step is required. We have ignored the beginning and the end of every training session to partially align the signals. Demonstrators unconsciously tend to keep the same speed during the motions. Surprisingly, throughout every session we get large variance in the duration of inactivity before actually executing the demonstration. We do the same in the end of each demonstration. After the alignment process, we have to resample our signals, in order to get a fixed number of points. The final training dataset for the GMM approach is in Figure 3 and can be compared with the data used in the CoSCo-HSOM (Figure 4).

We can point out three weaknesses of CoSCo-HSOM. First, there is some oscillation in the first steps of the reproduced gesture. As mentioned in Section II-C.2, the agent presents a reflexive behavior ($currentState \rightarrow nextState$)

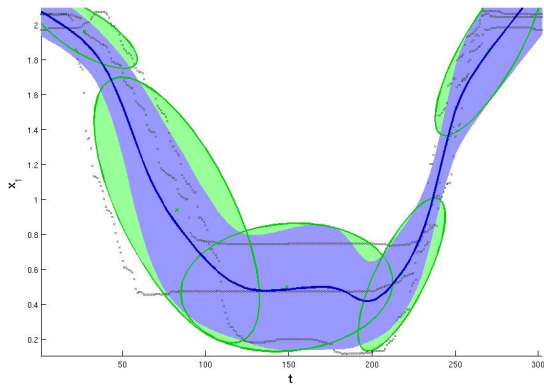


Fig. 5. Reproduction of a adflkjdf learned

in the very beginning, until it builds a short history. Even later though, we get sharp edges. This is virtually unavoidable, as of the nature of HSOM, where in practice detects the principal components of a gesture. That means we compress the gesture information and the output are the gesture's key poses. Finally, a consequence of that is the absorption of the time variant, making the algorithm perfect for gesture recognition, but "problematic" for gesture production. The controller has to implicitly decide the length of the gesture, while executing the transition between postures. Paradoxically, we take advantage of that "problem", which allows us to be more flexible on the reproduction of the motion. That means, we are able to execute the motion in varying speeds independently of the "expert"'s speed during the training session. Additionally, it allows incremental retraining from different experts with no prior knowledge of the gesture's temporal information. Depending on the configuration we use, we expect to further improve the results we get, and avoid any oscillations or spikes in the reproductions. In the accompanied video, it is clear that even with these oscillations the reproduced gesture remains smooth enough to be acceptable for execution.

In Figure 5, we present only the reproduction of the "Left Arm Shoulder Pitch" joint. It is obvious that the GMM approach, produces smooth motions and manages to remain smooth on the transition of one motion to another. It is also clear that the GMM algorithm manages reproduce gestures without losing any temporal information. The memory requirements are small as well, since we only need to save the Gaussian Mixture information (centers, covariance matrices). The drawback of this work though, is that if we were using raw data without doing any signal alignment [17] (the resampling process is unavoidable), we expect then that the reproduced motions would not be nicely constraint to get a meaningful reproduction through Gaussian Mixture Regression.

V. CONCLUSION

We have presented the CoSCo-HSOM algorithm, an hierarchical Self-Organizing map that manages to learn, and

reproduce gestures in humanoid and general purpose robots. A simple experiment helped us understand the effectiveness of our approach and its novel characteristics. We also made a comparative discussion against the Gaussian Mixture Model approach. Our future plans include an extensive analysis of the limits of this work and other domains where the CoSCo-HSOM could be potentially proven useful. We are confident that sensory-motor control applications will take advantage of this approach as well, and we expect to explore these areas.

VI. ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 231500.

REFERENCES

- [1] N. E. Sharkey, "Biologically inspired robotics," *Handbook of Brain Theory and Neural Networks*, 2002.
- [2] L. B. Cohen, H. H. Chaput, and C. H. Cashon, "A constructivist model of infant cognition," *Cognitive Development*, vol. 17, no. 3-4, pp. 1323-1343, 2002. [Online]. Available: [http://dx.doi.org/10.1016/S0885-2014\(02\)00124-7](http://dx.doi.org/10.1016/S0885-2014(02)00124-7)
- [3] S. Grossberg, *Neural networks and natural intelligence*. Cambridge, MA, US: The MIT Press, 1988.
- [4] J. Lampinen and E. Oja, "Clustering properties of hierarchical self-organizing maps," *Journal of Mathematical Imaging and Vision*, vol. 2, no. 2-3, pp. 261-272, 1992.
- [5] Y. Demiris and A. Billard, "Special issue on robot learning by observation, demonstration, and imitation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 2, pp. 254-255, 2007.
- [6] D. Gouaillier and P. Blazevic, "A mechatronic platform, the Aldebaran robotics humanoid robot," *32nd IEEE Annual Conference on Industrial Electronics, IECON 2006*, pp. 4049-4053, November 2006.
- [7] G. Pierris and M. Lagoudakis, "An interactive tool for designing complex robot motion patterns," *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, May 2009.
- [8] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39-42, 2004.
- [9] F. D. Libera, T. Minato, I. R. Fasel, H. Ishiguro, E. Pagello, and E. Menegatti, "A new paradigm of humanoid robot motion programming based on touch interpretation," *Robotics and Autonomous Systems*, vol. 57, no. 8, pp. 846-859, 2009.
- [10] A. Billard and R. Siegwart, "Robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 65-67, 2004.
- [11] S. Calinon and A. Billard, "Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm," in *ICML*, 2005, pp. 105-112.
- [12] N. Otero, A. Alissandrakis, K. Dautenhahn, C. L. Nehaniv, D. S. Syrдал, and K. L. Koay, "Human to robot demonstrations of routine home tasks: exploring the role of the robot's feedback," in *HRI*, 2008, pp. 177-184.
- [13] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.
- [14] A. Shimada and R. ichiro Taniguchi, "Gesture recognition using sparse code of hierarchical som," in *ICPR*, 2008, pp. 1-4.
- [15] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, vol. 37, no. 2, pp. 286-298, 2007.
- [16] S. Calinon, *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.
- [17] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *KDD Workshop*, 1994, pp. 359-370.