Learning Motor Control by Dancing YMCA

Rikke Amilde Lävlid

Learning Motor Control by Dancing YMCA

Rikke Amilde Løvlid

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway

Abstract. To be able to generate desired movements a robot needs to learn which motor commands move the limb from one position to another. We argue that learning by imitation might be an efficient way to acquire such a function, and investigate favorable properties of the movement used during training in order to maximize the control system's generalization capabilities. Our control system was trained to imitate one particular movement and then tested to see if it can imitate other movements without further training.

1 Introduction

Humanoid robots assisting humans can become widespread only if they are easy to program. This might be achieved trough learning by imitation, where a human movement is recorded and the robot is trained to reproduce it. However, to make learning by imitation efficient, good generalization capabilities are crucial. One simply cannot demonstrate every single movement that the robot is supposed to make.

How we want the agent to generalize depends on what we want the agent to do. When watching the demonstrator move, the robot can either learn to mimic the motion of the demonstrator or learn how the demonstrator acts in many situations, that is, extracting the intention of the movement. Mimicking the exact movement trajectory might be important when learning a dance movement, but this is less important when washing the dishes. Belardinell et al. taught a robot to extract salient features from a scene by imitating the gaze shifts of a human demonstrator [1]. Wood and Bryson used observations of an expert playing a computer game to make the agent learn what contexts are relevant to selecting appropriate actions, what sort of actions are likely to solve a particular problem, and which actions are appropriate in which contexts [2].

Learning by imitation is, in a sense, something in between pre programming the agent's control policy (i.e., the function that decides which action to choose in every situation), and letting the agent figure it out on its own through trial and error. According to a hypothesis in developmental psychology, learning to control ones own motor apparatus may be based on so called motor babbling, i.e., random exploration of joint angles [3, 4]. Other findings suggest that children use more goal directed movements [5].

We argue that imitation can be used in an efficient way in learning to master the motor apparatus. In this paper we investigate the features of the training movement required to make the suggested control system generalize to new movements, and illustrate how imitation can be used to make the agent train on movements that are most valuable in terms of future generalization capabilities.

2 Feedforward Control

The goal of the work presented here is to make the agent capable of moving its limbs to the positions it desires, that is, we want the agent to learn *feedforward control*. In feedforward control the future desired state is monitored and a motor command is issued to drive the system towards this state. We could call this proactive motor control. The purely reactive alternative is *feedback control*, where the state of the system is compared with the desired state of the system and adjustive motor commands are issued accordingly. Often both feedforward- and feedback control is needed. In our experiments we have used a feedback controller to train the feedforward controller.

We consider feedforward control as a modular process where the control policy, i.e., the function that maps the current state and the future goal to a motor command, is decomposed into a *planning stage* and an *execution stage*. The planning stage generates a desired trajectory. This can be realized by generating the whole desired sequence in advance, or through a next state planner. In the presented work planning is done by the demonstrator, and our focus is on the execution stage.

2.1 Realization of Feedforward Control with Internal Models

There are several ways to realize the execution stage in feedforward control, but most research regarding voluntary motor control shows a tendency towards the use of *internal models*. An internal model is a system that mimics the behavior of a natural process. In control theory, two types of internal models are emphasized, *forward models* and *inverse models*. A forward model predicts the outcome of an action (i.e., motor command). An inverse model represents the opposite process of calculating an action that will result in a particular outcome, the desired next state. Existence of internal models in the brain is widely accepted and there are many theories of how they are used and where they are located [6–8].

Forward models, inverse models and feedback controllers can be combined in different ways to calculate the desired motor command [9, 10, 6, 11]. The straightforward approach is to use only an inverse model. Since the input-output function of the inverse model is ideally the inverse of the body's forward dynamics, an accurate inverse model will perfectly produce the desired trajectory it receives as input. To acquire such an accurate inverse model through learning is, however, problematic. Kawato investigated different possibilities [9]. Among these, we have adopted the *feedback-error-learning* scheme, where a simple feedback controller is used together with the inverse model. The details are explained in section 3.2.

2.2 Implementation of the Inverse Model

In our control system, the inverse model was implemented as an *echo state network* (ESN) [12]. The basic idea with ESNs is to transform the low dimensional temporal input into a higher dimensional *echo state* by using a large, recurrent neural network (RNN), and then train the output connection weights to make the system output the desired information.

Because only the output weights are altered, training is typically quick and computationally efficient compared to training of other recurrent neural networks, and also simpler feedforward networks.

A typical task can be described by a set of input and desired output pairs, $[(i_1, o_1), (i_2, o_2), ..., (i_T, o_T)]$ and the solution is a trained ESN whose output y_t approximates the teacher output o_t, when the ESN is driven by the training input i_t.

Initially, a random RNN with the Echo State property is generated. Using the initial weight matrixes, the network is driven by the provided input sequence, $[i_1, i_2, ...i_n]$, where n is the number of time steps. Teacher forcing is used, meaning o_t is used instead of y_t when computing the state of the network at $t + 1$. The state of each node at each time step is stored in a state collection matrix, $\mathbf{M}$. Assuming $tanh$ is used as output activation function, $tanh^{-1}o_t$ is collected for each time step into a target collection matrix, $\mathbf{T}$.

If $\mathbf{W}^{out}$ is the weights from all the nodes in the network to the output nodes, we want to solve the equation $\mathbf{MW}^{out} = \mathbf{T}$. To solve for $\mathbf{W}^{out}$ we use the Moore-Penrose pseudoinverse; $\mathbf{W}^{out} = \mathbf{M}^{+}\mathbf{T}$.

Note that when the desired output is known, the network will learn the input-output function after only one presentation of the training sequence.

The input of the inverse model is the current state together with the desired next state, and the desired output is the desired motor command. The desired motor command is only known indirectly trough the desired position of the limbs. Generally, several motor commands may result in the same position of the limbs, and one does not want to bias the controller into choosing one specific solution. In sections 3.2 and 3.5 it is explained how an estimate of the desired motor command is used for teacher forcing and when generating the target collection matrix.

3 Learning the Inverse Model by Imitation

Our agent is implemented as a simple stick-man-simulator. After it has learned to imitate one movement, we want it to be able to imitate any movement presented by the demonstrator without further training. The input to the control system is always the current state and the desired next state (which is provided by the demonstrator). The goal is thus to learn the function mapping the current state and desired next state to the motor command, preferably with minimal effort.

3.1 The Movement Data

In the implementation of the experiments, we used a recording of the dance to the song YMCA by The Village People (se figure 1). The movement data was gathered with a Pro Reflex 3D motion tracking system by Axel Tidemann [13].

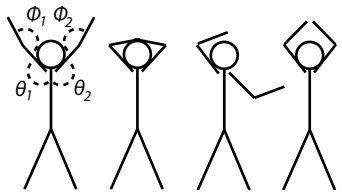

Fig. 1. The Movement is described with four angles; the two shoulder abduction angles $\theta 1$ and $\theta 2$ and the elbow extension angles $\phi 1$ and $\phi 2$.

The movement of each arm was described in two degrees of freedom, the angle between the arm and the body, i.e., the abduction angle θ, and the angle between the under- and upper arm, i.e., the extension in the elbow ϕ. Hence, the simulated robot was described by 4 degrees of freedom.

The YMCA movement was represented as a sequence of states, where each state t represents the four desired joint angles at time step t. The movement was manipulated in different ways to generate various training and testing sequences.

The goal of the control system is to produce motor commands that generate a movement where the state generated in each time step is as close as possible to the corresponding state in the desired sequence of states.

3.2 The Architecture

The control architecture that underlie the experiments is shown in figure 2. It consists mainly of an inverse model, but to achieve this model, a feedback controller is included during training.

At each time step t the control system receives as input the *desired next state* (i.e., the joint angles at time step $t+1$ in the training sequence) and the *current state*, and outputs a *motor command*, u.

During the training phase the feedback controller translates analytically the difference between the *desired current state* and the *actual current state* (i.e., state error) to a motor command, u_{error}. This motor error, the error done by the control system in the previous time step, is used to adjust the motor command for the current time step. This works as an approximation to teacher forcing because the only connection from the output nodes back to the network is through the plant, providing the current state input at the next time step. How much the feedback controller is able to influence the motor command depends on the *feedback gain*, K, by letting $u_{feedback} = K * u_{error}$. Note that during testing $K = 0$. The motor command $u_{feedback}$ is added to the motor command produced by the inverse model, and the result is sent to the plant, i.e., the robot simulator.

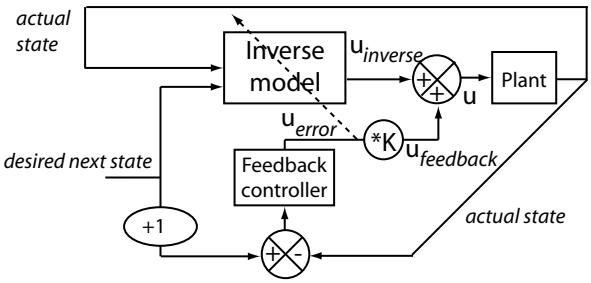

Fig. 2. The control architecture. Delays are shown with ellipses, i.e., the desired next state is delayed one time step, now representing the desired current state, before given as input to the feedback controller. An arrow across a model represents training, and u is a motor command.

3.3 Hypotheses

First, we need to verify that the control system presented is able to imitate novel movements when trained to perform one movement. This would imply that the system has learned at least parts of the function that computes the motor command needed to move the agent from it's current position to the desired next position. Second, we investigate further what properties the training movement must possess in order to make the system generalize to any movement in the state space. Our aim can be conveyed through the following tree hypotheses:

Hypothesis 1: *When training on imitating one movement, the control system does not only learn to mimic that movement, but learns at least parts of the function mapping a current and a desired state to a motor command, which will make it able to imitate other movements without training.*

Hypothesis 2: *In order to learn to control one particular degree of freedom, it has to be included in the training movement.*

Hypothesis 3: *When trained on synchronous movements, i.e., the movement of the two arms are equivalent, mirroring each other, the control system is only able to imitate synchronous movements. Training on movements where the limbs follow different trajectories is necessary in order to make the control system generalize to all movements.*

3.4 Experiments

To generate different training and testing sequences the YMCA movement was manipulated in different ways. The movements were **YMCA** (the whole YMCA movement), **Y** (only the Y movement, moving back to start position by reversing the Y motion), **Y pure** (a manipulated version of the Y movement, where all movement in elbow angle is removed), **YM** (only the YM movement, moving back by reversing the YM sequence), **right arm mirror** (both arms does the movement of the right arm in the YMCA movement, making the arms mirror

each other) and **left arm mirror** (similar to *right arm*, but now both arms moves as the left arm in the YMCA movement).

3.5 Training

The training procedure was organized in epochs and cycles, where one cycle is one full temporal presentation of the training motion. In each epoch we ran seven cycles. First, we re-initialized the network by setting the internal states of the network to zero and run one cycle without updating the output weights. Subsequently, the training sequence was presented five times with enabled learning. The output connections were then adapted after each complete cycle. A final cycle was used to estimate the performance error on the training sequence while learning was disabled. The training was run for 150 epochs.

Multiple training epochs was needed because perfect teacher forcing could not be provided. To make the estimate of the desired motor command, u, as good as possible, the feedback controller should provide less influence as the inverse model gets more accurate. This was ensured by decreasing the feedback gain, K, by 10% each epoch.

The output from the feedback controller was also used when calculating the target collection matrix. Because we had batch learning, the motor command was stored and the target motor command u_{target} was calculated by using the u_{error} provided in the next time step, $u^t_{target} = u^t + u^{t+1}_{error}$ (because u^t_{error} reflects the error done at time step $t-1$).

The inverse model had 8 input nodes, 1000 nodes in the internal layer and 4 output nodes. The ESN had spectral radius $\alpha = 0.1$ (determining the length of the memory with increasing $\alpha \in [0, 1]$) and noise level $v = 0.2$ (effectively adding 10% noise to the internal state of the network). The inputs and outputs where scaled to be in the range [-1,1]. Normal-distributed noise with standard deviation 0.01 was added to the sensory signal from the plant. The system was implemented in MatLab, including the simple stick-man-simulator used as plant.

3.6 Testing

After the inverse model was trained to imitate one movement, we wanted to test whether it could imitate other movements without training. This was done by changing the desired sequence and run the network for one additional epoch with only two cycles, the initialization cycle and the evaluation cycle.

During training the feedback gain was decreased to ~ 0, and the feedback controller was thus removed from the control system during testing.

4 Results

This section summarizes the results of the experiments. The results verifying hypotheses 1, 2 and 3 are described in section 4.1, 4.2 and 4.3 respectively. We have illustrated the results by plotting the shoulder abduction angle, θ against

the elbow extension angle, ϕ for each arm. The temporal dimension is not plotted because all discrepancies between actual and desired movement could be seen as spatial errors, the timing turned out to not be a problem in any of the experiments. Figure 3 shows a plot of the whole YMCA movement where the different parts of the movement is separated by the use of different markers.

4.1 Does the Control System Generalize?

The initial experiment was to train the control system to imitate the whole YMCA movement and see if it was able to produce the correct movements when tested on the other movements described in section 3.4. We also tested whether the control system would generalize to different speeds. The control system managed all these tests without significantly more error than when imitating the trained movement. We conclude that when trained with the whole YMCA movement, the control systems learned the complete mapping from current- and desired state to motor command in the state space.

4.2 Training All Degrees of Freedom

Does all degrees of freedom, used in the testing sequence, need to be included in the training sequence? To test this hypothesis the control system was trained on *Y pure* and tested on *YM*, see figure 4. The control system is clearly not able to utilize the joint it has not trained to use. To find out how small perturbations in the joint is sufficient for generalization, we tried training on Y and testing on *YMCA*. As figure 5 illustrates, small angular changes in the joint during training, makes it possible to generalize to larger changes during testing, but not large enough to perform *YMCA* without large errors.

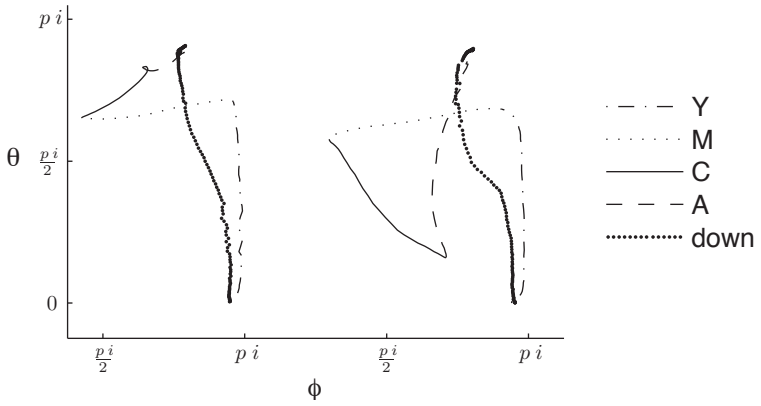

Fig. 3. The whole YMCA movement. The shoulder abduction angle, θ is plotted against the elbow extension angle, ϕ for each arm. The left graph shows the movement of the right arm and the right graph the movement of the left arm. Each part of the movement is plotted with a different marker to distinguish them from each other.

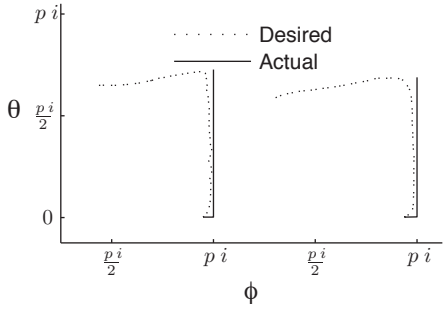

Fig. 4. The figure shows the trajectory produced by the two arms when the control system is trained on *Y pure*, and tested on *YM*. In the training sequence, *Y pure*, there is no movement in the elbow joints, and thus the network is not able to utilize these during testing.

4.3 Synchronous and Asynchronous Movements

To test whether the system can generalize to asynchronous movements when trained with a pure synchronous movement, we used *right arm mirror* and *left arm mirror* as training sequences and tested on *YMCA* (see figure 6).

The system clearly does not generalize to asynchronous movements; the movement of the arms were more or less symmetric even though the test sequence is not. The movement of each arm was an average of the movements of the two arms in the desired sequence. As a consequence the results are practically identical when the control system was trained with *right arm mirror* compared to when trained with *left arm mirror*.

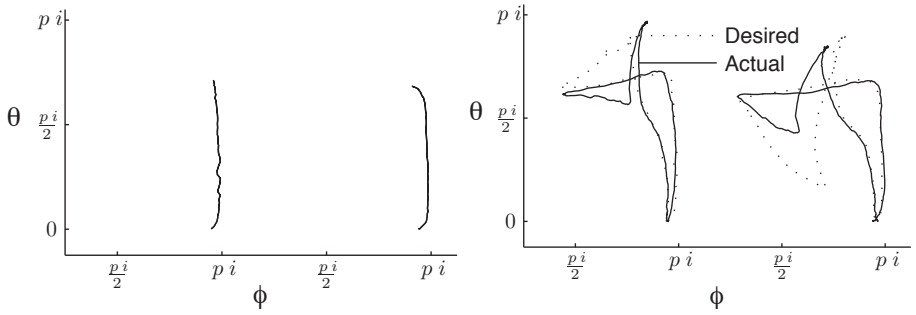

Fig. 5. The left figure illustrates the *Y* movement. Notice that there is hardly any motion in the elbow. The figure to the right shows the result when the control system trained on movement *Y* is tested on movement *YMCA*. The control system is able to generate more motion in the elbow joints than it learned during training. However, it is not able to produce the *YMCA* movement without large errors.

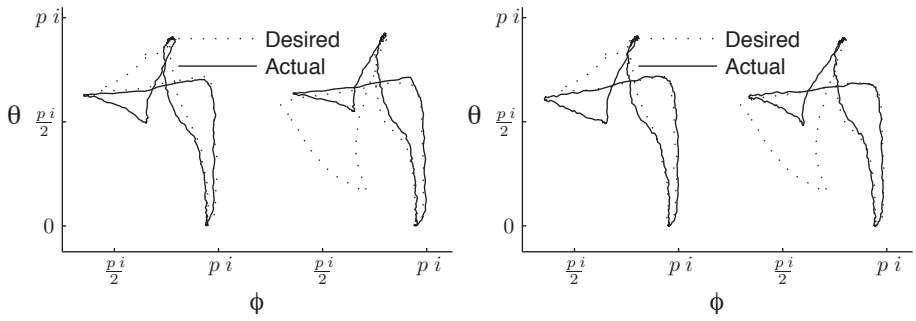

Fig. 6. The figures illustrates the results when trying to do an asynchronous movement when trained to do a synchronous one. The control system produces a synchronous movement that is the average of the desired movement of the two arms. In the figure to the left the system was trained on *left arm mirror* and in the right figure, *right arm mirror*. Both were tested on *YMCA*.

Remember that the opposite problem of imitating a synchronous movement when trained with an asynchronous movement does not pose any problem. When trained with the asynchronous movement *YMCA*, the system was able to generate all the movements without difficulty.

5 Discussion and Conclusion

Our aim was to use imitation to efficiently learn the inverse kinematics model of our robot simulator. We showed that when trained to imitate one movement, the control system has not only learned to imitate that particular movement, but is able to imitate novel movements without further training. This means that the system has learned at least parts of the desired inverse model, verifying hypothesis 1.

Our second hypothesis envisages that in order to learn to control one particular degree of freedom, it has to be included in the training movement. We showed this to be true. In addition, our results suggest that the control system does not have to train on the whole range of motion for each degree of freedom in order to generalize to all movements. This is important when we want to train the inverse model with minimal amount of effort.

Hypothesis 3 suggests that asynchronous movements are harder than synchronous movements, and that the control system will not be able to produce different motor commands for the two arms if it has not been trained to do so. For humans it is indeed true that it is easier to move the limbs synchronously. It is still very interesting that we get the same results for this control system, and interesting to see that a system trained to produce a synchronous movement and asked to generate an asynchronous movement provides the best solution it is

able to, namely the average between the desired movement of the left and right arm.

Our findings suggests that imitation may be used as an efficient method to learn the inverse model, because one can choose the training sequence optimally, as opposed to exploration without guidance. This conclusion is supported by Rolf et. al. who suggests the use of goal directed exploration in contrast to motor babbling [14].

Further work should include more complex movements with larger degrees of freedoms where one target position can be reached through different motor commands. In addition more systematic evaluation of efficient training movements should be conducted.

References

1. Belardinelli, A., Pirri, F.: Bottom-up gaze shifts and fixations learning by imitation. IEEE Trans Syst Man Cybern B Cybern **37**(2) (2007) 256–271
2. Wood, M.A., Bryson, J.J.: Skill acquisition through program-level imitation in a real-time domain. IEEE Trans Syst Man Cybern B Cybern **37**(2) (2007) 1083–4419
3. Meltzoff, A.N., Moore, M.K.: Explaining facial imitation: A theoretical model. Early Development and Parenting **6** (1997) 179–192
4. Demiris, Y., Dearden, A.: From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In: Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. (2005) 31–37
5. von Hofsen, C.: An action perspective on motor development. TRENDS in Cognitive Sciences **8**(6) (2004) 266–272
6. Wolpert, D.M., Miall, R.C., Kawato, M.: Internal models in the cerebellum. Trends in Cognitive Sciences **2**(9) (1998)
7. Shadmehr, R., Krakauer, J.W.: A computational neuroanatomy for motor control. Experimental Brain Research **185**(3) (2008) 359–381
8. Davidson, P.R., Wolpert, D.M.: Widespread access to predictive models in the motor system a short review. Journal of Neural Engineering **2**(3) (2005) 313–319
9. Kawato, M., Gomi, H.: The cerebellum and vor/okr learning models. TINS **15**(11) (1992)
10. Mehta, B., Schaal, S.: Forward models in visuomotor control. Journal of Neurophysiology **88**(2) (2002) 942–953
11. van der Smagt, P., Hirzinger, G.: The cerebellum as computed torque model. In Howlett, R., Jain, L., eds.: Fourth International Conference on Knowledge-Based Ingelligent Engineering Systems & Applied Technologies. (2000)
12. Jaeger, H.: A tutorial on training recurrent neural networks, covering bppt, rtrl, and the echo state network approach. Techreport, Fraunhofer Institute for AIS (April 2008)
13. Tidemann, A., Öztürk, P.: Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA. In: IEA/AIE. Volume 4570 of LNCS., Springer (June 2007) 291–302
14. Rolf, M., Steil, J.J., Gienger, M.: Efficient exploration and learning of whole body kinematics. In: IEEE 8th International Conference on Development and Learning. (2009)