In Visser, Ribeiro, Ohashi, and Dellaert, editors, *RoboCup-2007*,
Springer Verlag, 2008.

A Neural Network-Based Approach to Robot Motion Control

Uli Grasemann, Daniel Stronger, and Peter Stone

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
{uli,stronger,pstone}@cs.utexas.edu

Abstract. The joint controllers used in robots like the Sony Aibo are designed for the task of moving the joints of the robot to a given position. However, they are not well suited to the problem of making a robot move through a desired trajectory at speeds close to the physical capabilities of the robot, and in many cases, they cannot be bypassed easily. In this paper, we propose an approach that models both the robot's joints and its built-in controllers as a single system that is in turn controlled by a neural network. The neural network controls the entire trajectory of a robot instead of just its static position. We implement and evaluate our approach on a Sony Aibo ERS-7.

1 Introduction

Commercially available robots like the Sony Aibo usually come with built-in controllers that are designed to allow precise control over the robot's joint positions. In many applications, however, the goal is not simply to make the robot move to a given position, but rather to make it execute a given motion, i.e. to control the robot's position at all points in time. Furthermore, tasks like robot soccer, in which speed is an important factor, require the robot to execute motions like walks or kicks both precisely and at speeds close to the physical limits of the robot's effectors. The built-in controllers that come with robots like the Aibo are not designed for this task.

Most approaches that would allow precise control over a robot's motion require exact knowledge of all properties of the joints and motors involved, as well as the ability to bypass the built-in controllers and access the robot's effectors directly. Inexpensive, commercially available robots like the Sony Aibo usually do not meet these conditions.

In this paper, we explore an alternative approach to the problem, which uses neural networks that learn to predict the commands that are necessary in order to make the robot execute a predefined motion. The robot and its built-in joint controllers are both treated as part of system to be modeled. We implement and evaluate the proposed approach on a Sony Aibo ERS-7[1].

2 Background

[1] http://www.aibo.com

Standard control theory [1] focuses on the task of finding a controller H that, given an observation of the state x of a system, provides an appropriate action at every time step such that the system eventually reaches a given target state. This paper considers the goal of enabling a robot to accurately execute a desired movement: a trajectory through its state space over time. The classical approach to robotic trajectory planning involves controlling the forces or torques exerted by the joints directly [2]. Applying this technique requires the parameters and specifications of the robot to be known, as well as low-level access to the robot's effectors. Inexpensive, commercially available robots like the Aibo usually meet neither of these conditions.

Fig. 1. A Sony Aibo ERS-7. The arrows point to the shoulder (1), abductor (2), and knee joint (3) of the Aibo's right front leg.

On the Sony Aibo robot, all control of the joints goes through the robot's API, which at the lowest level uses PID control [3]. However, previous work [4] suggests that each joint (at least on the Aibo ERS-210) can not be completely understood based on the theory of PID control. Nevertheless, because it is only possible to issue commands to the robot's joints through the PID controller, this paper considers that controller as part of the dynamical system, and therefore part of the problem. If the target is close to the actual position, for example, it will not move at maximum speed, even if maximum speed is required at the present part of the trajectory.

Several alternative approaches are commonly used to make a robot execute a given movement more reliably. For example, one way to increase precision is to slow down the movement of the robot so that the angle speeds involved are well below the maximum angle speeds possible. Another possibility is to search for a set of parameters used to create a sequence of angle requests, where an end goal like overall robot speed is used as a reward function [5, 6].

Ideally, we would like some kind of an equivalent of classical controllers for motion control. As in standard control theory, there are two basic options. First, the equivalent of an open-loop controller would be a functional H_{open} that maps a desired trajectory T through the system's state space onto a sequence of appropriate actions U for any given time: $H_{open} : T \mapsto U$. This sequence can then be used on the robot in order to achieve the desired movement. Second, the equivalent of a feedback-controller would take as its input the present time t, the target trajectory T, and the current observation y of the robot's state x. Its output would be an appropriate action u that keeps the robot's motion sufficiently close to the target trajectory at any time:

$$H_{feedack} : (T, t, y) \mapsto u$$

In the real world, defining and finding such functionals is simplified by the fact that both time and the state space of a robot are effectively discrete, and the dependencies between the motor commands and the robot's trajectory are highly local. This allows us to define H_{open} in terms of a function h_{open} that maps a finite neighborhood $\pm l$ of T around a time t onto a single action u at the same time t: $h_{open} : (T(t-l), \ldots, T(t+l)) \mapsto u$. H_{open} is then defined by computing h_{open} for every discrete time step of T. For example, Stronger and Stone [4] construct such a function h_{open} for piecewise-linear trajectories by first constructing an empirical joint model of a Sony Aibo ERS-210, and then inverting that model to obtain h_{open}. By contrast, the approach described in this paper uses a neural network to obtain such functions directly using data acquired from a robot.

Finally, a wide range of previous work has also used neural networks to control robotic motion. Lewis et al. [7] show how neural networks can be used to approximate nonlinearities in the robot's dynamics. This method can be used for trajectory planning, but doing so requires direct control of the robot's motors, which is not available on many commercial robots. On a Sony AIBO, Billard and Ijspeert [8] use a neural network to generate qualitative variations on a type of motion, such as different gaits for walking. Angulo et al. [9] apply neuroevolution to a Central Pattern Generator (CPG) to demonstrate the emergence of a walking behavior. To the best of the authors' knowledge, these approaches have not been applied to the task addressed in this paper: performing accurate motion along an arbitrary trajectory.

3 A Neural Network-Based Approach to Robot Motion Control

The basic idea behind our approach to robot motion control is simple: We use a neural network to predict which motor commands will cause the robot to execute a given movement. The robot's joints and their controllers together form the system we are trying to control.

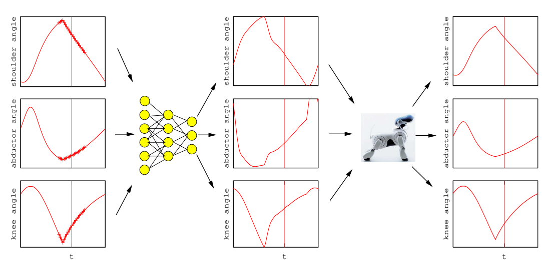

Fig. 2. The basic structure of the proposed motion controller at time t. A neural network maps a neighborhood of the target trajectory around t onto appropriate motor commands at time t. The motor commands are used by the Aibo and result in a motion close to the target trajectory.

Figure 2 illustrates the structure of our approach: A neural network maps a finite neighborhood of the target trajectory around the present time step onto a set of motor commands that is supposed to keep the robot on that trajectory. Computing the output of the neural network for each time step of a target trajectory gives a sequence of angle requests that can then be used by the robot to execute the desired motion. Note that the neural network plays the role of the function h_{open} defined in the previous section, implicitly defining an open-loop motion controller H_{open}.

The neural network, which represents the inverse of the system in question, can be learned directly from raw data: All that is needed is a sequence of angle requests U and the resulting movement T. For each time t, the neighborhood $T(t-l) \dots T(t+l)$, together with the angle request $U(t)$, forms a training pattern for the network.

4 Experiments

The experiments reported in this paper were conducted using a Sony Aibo ERS-7, a commercially available four-legged robot with 17 degrees of freedom. All joints are equipped with PID controllers that cannot be bypassed to control the Aibo's movements on a lower level, and the precise specifications of both the Aibo's effectors and of the PID controllers are not documented. The ERS-7 has sensors on each joint that allow precise recording of the actual effects of any motion command. The motion commands are given in the form of one angle requests for each joint every 8ms. We used this maximum frequency in all experiments.

For the reported results, we focused on the task of controlling the Aibo's right front leg while the robot was not touching the ground. Figure 1 shows an ERS-7 and points out the joints involved in the reported experiments.

The first set of experiments served two separate purposes. First, it aimed to establish that the approach described in the last section leads to a significant improvement over just using the raw trajectory as motion commands. The experiment's second purpose was to find out whether a single neural network model of all joints involved performs better than having separate neural networks for each joint.

The first step was to create the neural network models of the Aibo's joints. As mentioned before we focused on controlling the Aibo's right front leg, which has three degrees of freedom: The shoulder, the abductor, and the knee (see Figure 1.) The Aibo was held in the air such that the leg never touched the ground.

4.1 Experiment I

We acquired training data for the neural networks by first creating a random continuous sequence of angle requests, then running those requests through an Aibo and recording the resulting movements using its sensors. Comparing the original angle requests and the resulting target trajectory in Figure 3 should give an impression of the kind of data used, although the data shown there were not part of the training set. Note how the actual trajectory lags behind the angle requests used to create it. Using about 80 seconds of training data, we then trained two different motion controllers for the Aibo's front leg: The first controller was intended to model all three joints at the same time using one neural network; the second controller used a separate network for each joint.

The input and output of the neural networks were exactly as described in section 3. Based on the estimated time it takes the Aibo to react to motor commands [4], we chose ± 10 time steps as the size of the neighborhood on the target trajectory. This meant that the single-network model had 60 inputs (three joints $\times$ 20), and three output nodes (one for each joint.) The networks that modeled a single joint had 20 input nodes and one output node.

All networks were fully connected feedforward-networks with one hidden layer. The size of the hidden layer was the same as the input layer. The networks were trained for 2000 epochs using standard backpropagation with a small momentum term (0.2). The training rate was 0.1 for the first 1000 epochs, and 0.05 for the rest of the time. We used SNNS (the Stuttgart Neural Networks Simulator [10]) to create and train the networks.

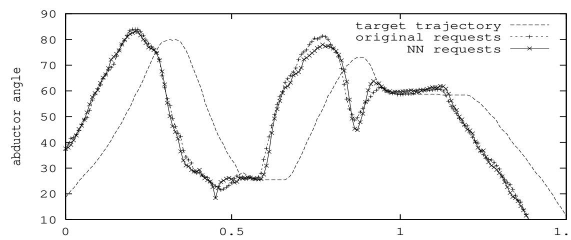

Fig. 3. Part of the test trajectory used in the first experiment (dotted line). Compare the original angle requests used to create the trajectory with the angle requests output by a neural network controller.

We then created a fresh sequence of random angle requests, and recorded the resulting movements of the Aibo's leg. Using these movements as a target trajectory, we used both neural network controllers independently to try and replicate the target trajectory on the Aibo.

In order to establish a reasonable baseline with which to compare our results, we also used the target trajectory as angle requests, after shifting it back by 12 time steps to allow for the lag. This is the equivalent of a controller that models only the time lag of the Aibo's joints. Additionally, we ran the original sequence of angle requests through the Aibo again, to find a practical upper performance limit due to motor and sensor precision.

Figure 3 shows part of the target trajectory for the Aibo's abductor joint, together with the original angle requests used to create it, and the angle requests that the single-network model thinks will reproduce the target trajectory. It seems like the requests created by the neural network stay reasonably close to the original.

Figure 4 compares the trajectories controlled by the two neural network controllers to the baseline trajectory. Both neural network controllers perform visibly better than the baseline, especially at sharp turns in the trajectory. Figure 5 compares the two neural network controllers. The height of each bar is the average Euclidean distance of the Aibo's foot from the target trajectory. The leftmost bar is the upper performance limit established by using the original set of angle requests again, and comparing the result to the target trajectory. The two bars in the center belong to the two neural network controllers. The bar on the right is the baseline error. The error bars denote the 95% confidence intervals for the distance averaged over 12 seconds. All differences are statistically significant ($p < .05$).

Figure 5 confirms the earlier impression that both neural network controllers perform well above the baseline. In fact, they both reduce the average error by more than half. Also, for the single network controller, the average distance from the target trajectory is less than twice the upper performance limit defined by the Aibo's motor and sensor precision. Overall, our first experiment showed that both neural network controllers perform significantly above the baseline level, and the single network-controller is able to exploit the additional information it receives to outperform the controller using separate networks.

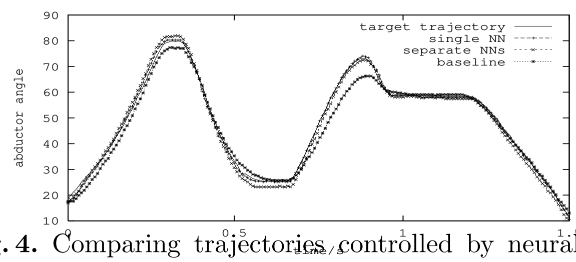

Fig. 4. Comparing trajectories controlled by neural networks with the baseline trajectory. Both neural network controllers perform visibly better than the baseline.

4.2 Experiment II

The second experiment also had two objectives. The first was to find out if the open-loop architecture chosen for the present implementation is able to handle trajectories outside the physical limits of the robot. Such trajectories are usually created to fool the built-in controllers into moving the joints faster than they would ordinarily, and would therefore be unnecessary given a working motion controller. However, it would still be useful to have a motion controller that, given a trajectory outside the physical constraints of the Aibo, creates the closest possible trajectory within the constraints.

The second objective was originally to make a quantitative comparison between our model and the analytical model used by Stronger and Stone for the same task [4]. Since the results reported there were obtained using an earlier model of the Aibo, we attempted to implement the model on the new Aibo in order to make a quantitative comparison possible.

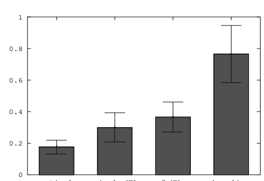

Fig. 5. The average Euclidean distance from the target trajectory (in cm) achieved by the neural network controllers are shown in the middle two bars.

However, early experiments revealed that the joint dynamics of the Aibo ERS-7 are sufficiently different from the earlier model as to make a direct implementation impossible. When the requested trajectory for a leg joint was set to a step function, the different joints exhibited qualitatively different and surprisingly erratic behaviors. The angle speeds changed unpredictably over time, and the joints' behavior did not appear to fall within the parameters of Stronger and

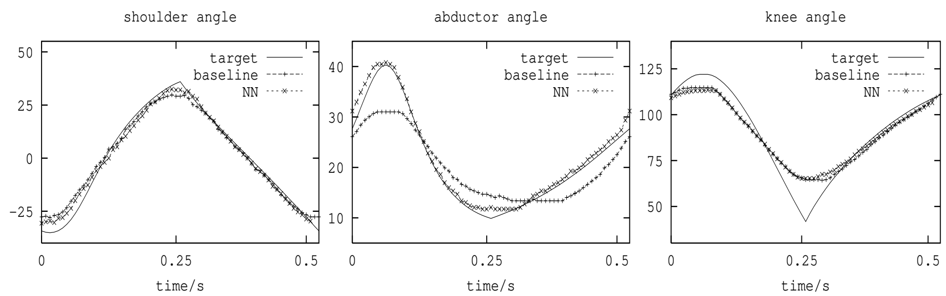

Fig. 7. The resulting angle trajectories for the shoulder, abductor, and knee joint. The shoulder and especially the abductor show improvement over the baseline trajectory.

Stone's model. We believe that this in itself is a strong argument for an adaptive and more flexible approach to joint modeling.

Figure 6 shows the target trajectory used in this experiment. It is a half-ellipse with a period 65 timesteps, and could be realistically used for a fast walk on the Aibo ERS-7.

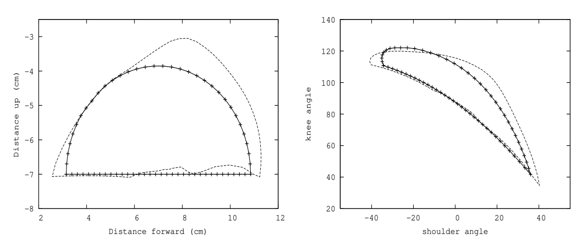

Fig. 6. The test trajectory used for the second experiment, in Cartesian coordinates (left), and in the Aibo's joint angle coordinates (right.) The dotted lines are angle requests created by a neural network controller for this trajectory.

We used the single-network controller trained in the last experiment to try and reproduce the target trajectory on the Aibo. Figure 7 shows the resulting angle trajectories for the three joints involved. Like in the last experiment, the baseline curve was obtained using a controller that only compensates for the time lag between an angle request and the resulting motion. The trajectories of the shoulder and especially the abductor clearly show improvement over the baseline curve, while the trajectory for the knee is more or less the same as the baseline.

It would be reasonable to expect a corresponding improvement of the trajectory of the Aibo's foot in Euclidean space. However, no such improvement was observed. The average distance of the Aibo's foot from the target trajectory is about 8.5mm both for the baseline trajectory and for the one controlled by the neural network. This discrepancy can be understood as follows. When the neural network achieves an improvement over the baseline, the large knee angles moved the Aibo's foot close to the rotational axis of the abductor joint, which made the improvement in the abductor angle irrelevant in Euclidean space.

Notably, the neural network-based controller degraded gracefully when presented with a physically impossible trajectory, since the controller used to create the baseline curve still performs much better than using the raw target trajectory as angle requests.

5 Conclusion

This paper introduced a neural network-based approach to robot motion control. Using data recorded on a physical robot, we trained neural networks to predict which angle commands are necessary to make a robot execute a given movement. The built-in controllers for the robot's joints were treated as part of the system to be modeled and controlled.

We conducted two experiments, using a popular commercially available robot, the Sony Aibo ERS-7, as our experimental platform. The first experiment showed that the proposed approach is indeed able to bring a robot's motions significantly closer to the desired trajectory. In the second experiment, the neural network-controller failed to produce equally good results, but was nevertheless shown to degrade gracefully when presented with a target trajectory outside the robot's physical constraints.

Acknowledgements

This research is supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545. The authors thank Peggy Fidelman and Nate Kohl for useful discussions.

References

1. Bubnicki, Z.: Modern Control Theory. Springer (2005)
2. Sciavicco, L., Siciliano, B.: Modeling and Control of Robot Manipulators. McGraw-Hill Companies, Inc. (1996)
3. Johnson, M., Moradi, M., eds.: PID Control: New Identification and Design Methods. Springer (2005)
4. Stronger, D., Stone, P.: A model-based approach to robot joint control. In Nardi, D., Riedmiller, M., Sammut, C., eds.: RoboCup-2004: Robot Soccer World Cup VIII. Springer Verlag, Berlin (2005) 297–309
5. Kim, M.S., Uther, W.: Automatic gait optimisation for quadruped robots. In: Australasian Conference on Robotics and Automation, Brisbane (2003)
6. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2004)
7. Lewis, F., Jagannathan, S., Yesildirek, A.: Neural Network Control of Robot Manipulators and Nonlinear Systems. Taylor & Francis (1999)
8. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for motor control in a quadruped robot. In: International Joint Conference on Neural Networks. Volume 6. (2000)
9. Angulo, C., Tellez, R., Pardo, D.: Emergent walking behaviour in an aibo robot. The European Research Consortium for Informatics and Mathematics (2006)
10. Zell, A., Mache, N., Hbner, R., Mamier, G., Vogt, M., Schmalzl, M., Herrmann, K.U.: Snns (stuttgart neural network simulator). In Skrzypek, J., ed.: Neural Network Simulation Environments. Kluwer Publishers (1993)