Kinesthetic Bootstrapping: Teaching Motor Skills to Humanoid Robots through Physical Interaction

Heni Ben Amor, Erik Berger, David Vogt, and Bernhard Jung

VR and Multimedia Group, TU Bergakademie Freiberg, Freiberg, Germany
{amor,bergere,vogt3,jung}@mailserver.tu-freiberg.de

Abstract. Programming of complex motor skills for humanoid robots can be a time intensive task, particularly within conventional textual or GUI-driven programming paradigms. Addressing this drawback, we propose a new programming-by-demonstration method called *Kinesthetic Bootstrapping* for teaching motor skills to humanoid robots by means of intuitive physical interactions. Here, "programming" simply consists of manually moving the robot's joints so as to demonstrate the skill in mind. The bootstrapping algorithm then generates a low-dimensional model of the demonstrated postures. To find a trajectory through this posture space that corresponds to a robust robot motion, a learning phase takes place in a physics-based virtual environment. The virtual robot's motion is optimized via a genetic algorithm and the result is transferred back to the physical robot. The method has been successfully applied to the learning of various complex motor skills such as walking and standing up.

1 Introduction

Research in robotics and AI has lead to the emergence of increasingly complex anthropomorphic robots, such as humanoids and androids. In order to be meaningfully applied in human inhabited invironments, anthropomorphic robots need to possess a variety of physical abilities and skills. However, programming such skills is a labour and time intensive task which requires a large amount of expert knowledge. In particular, it often involves transforming intuitive concepts of motions and actions into formal mathematical descriptions and algorithms. Even in GUI-based programming environments where complex robot movements are specified as sequences of robot postures defined in the graphical user interface, much time is usually spent for parameter tweaking. Due to their relatively large number of degrees of freedom, this process becomes particularly cumbersome for the case of humanoid robots. To reduce the complexity of this task, more natural and intuitive approaches to programming robot skills are called for.

This paper presents a new programming-by-demonstration method for bootstrapping robotic motor skills through kinesthetic interactions. A human teacher instructs the robot by manually moving the robot's joints and body to postures that approximate the intended movement. Then, an automatic optimization

B. Mertsching, M. Hund, and Z. Aziz (Eds.): KI 2009, LNAI 5803, pp. 492–499, 2009.

phase takes place during which the robot learns a motor skill that still resembles but also compensates for likely imperfections of the demonstrated movement. This learning phase makes use of a physics-based virtual enviroment, where a large amount of movement variations can be tried out very quickly without the need for human intervention. In this way, the Kinesthetic Bootstrapping method introduced in this paper both simplifies and reduces the time for programming of robotic motor skills.

2 Related Work

The work presented in this paper can be regarded as a variant of imitation learning. In the context of robotics, the goal of imitation learning is to allow human teachers to program robotic agents by conveying a demonstration of the desired behavior. This is often realized using expensive motion capture or virtual reality techniques in order to record the example motions. In [1] the walking gait of a human demonstrator is first recorded through motion capture and then adapted for imitation by a small humanoid robot. However, such approaches need to tackle the *correspondence problem*: the question of how to map the joint information of the user onto the robot's body [2]. A variation of this approach can be found in [3]. Here, actions are recorded while the human demonstrator is interacting with a virtual reality environment using a data glove. Recently, more intuitive approaches to programming by demonstration have been pursued. In [4], direct kinesthetic interaction with humanoid robots has been used for teaching manipulation skills. In this approach, the instructor has to repeatedly convey demonstrations and provide corrective feedback to the learning robot. A similar approach has been used by Tani et al. [5] to encode demonstrated behaviors using recurrent neural networks. In both papers the imitated behaviors where limited to the upper body of the robot and did not involve complex, dynamic motions. In contrast to these works, the approach introduced in this paper works even with a *single* example demonstration but can also deal with several examples. The robot can adapt and improve the provided example autonomously, without relying on any further interaction with the instructor. Further, providing the example motion through a kinesthetic modality, allows us to increase the naturalness of the interaction. This is closely related to "physical programming languages" [6] found in human-computer interaction research.

3 Kinesthetic Bootstrapping

When learning a new physical skill, children are often supported by their parents. This allows to transmit knowledge on how to solve the task at hand and, thus, overcome learning barriers. In these situations, kinesthetic interactions serve as a communication channel between the parent and the learning child. The bodily experience resulting from these interactions helps to reduce the amount of time needed for acquiring the skill. Still, the child has to go through an unassisted learning phase in order to fully master the skill. Kinesthetic Bootstrapping

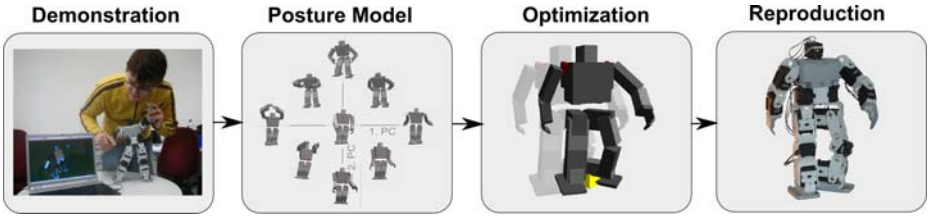

Fig. 1. Overview of the Kinesthetic Bootstrapping approach. After kinesthetic interaction, a posture model is created. A simulator is used to optimize the demonstrated skill. The result is then applied on the real robot.

applies the same principle to the programming of humanoid robots. A human conveys a demonstration of the task at hand through kinesthetic interactions. The bodily experience allows the robot to draw important information on the task at hand. This information is used to "bootstrap" the robot's knowledge, which is then used in a learning phase to reproduce the skill without any assistance. Figure 1 shows an overview of the learning approach used in Kinesthetic Bootstrapping.

First, the teacher moves the joints of the robot in order to convey a demonstration of the intended motion or behavior. This is done continuously without relying on keyframes or another kind of discretization. During the demonstration, the motor configurations of the robot are recorded with a frequency of 20 Hz. The robot used in this study is a *Bioloid* robot with 18 servo-motors (i.e. 18 degrees-of-fredom). In each step, the state of each of all servo-motors is recorded, resulting in an 18-dimensional posture vector p. Once the user finishes the demonstration, all posture vectors are collected in order to compute a low-dimensional posture model of the skill. Next, using the extracted posture model, different variations of the skill are evaluated. This is done in a physics-based virtual reality simulation of the robot. The simulator allows us to optimize the motion without harming the robot hardware and without reyling on human assistance. In particular, when the demonstrated motion is very dynamic, such as a standing up motion, it is important for the robot to learn how to account for the external (stabilizing) forces previously applied by the human teacher. Once the optimization phase is finished, the learned motion is transfered to the physical robot and replayed outside of the simulation. In the remainder of this section, we will explain each phase of the learning process in more detail.

3.1 Simulator

As mentioned above, learning and adaptation of the demonstrated skill is performed in a physics-based virtual reality simulator. The simulator is based on the Open Dynamics Engine (ODE) and contains a precise model of the Bioloid humanoid robot. For calibrating the model, each motor is automatically moved by the calibration software and the time needed to reach a given configuration by the real and simulated robot is measured. The difference between the two

time values, i.e. the discrepancy between the real and simulated world, is used to adapt the values of the low-level PID controller in simulation, so as to better fit the movements of the real robot. An important feature of this simulator is an abstraction layer for the control of the robot. This layer allows it to control the real or simulated robot or both using the same interface. Thus, the user can always decide whether to apply the current program in reality or simulation. During the learning phase, the simulator is used for reproduction of different variations of the originally demonstrated motion. In a trial-and-error fashion each variation is executed by the virtual robot, evaluated and the result used for further optimization.

3.2 Low-Dimensional Posture Models

The kinesthetically recorded demonstration can be regarded as a template from which important information about the skill in mind is inferred. More precisely, the demonstration is used to compute a model, from which new variations of the skill can be synthesized. In the following, such models will be referred to as *low-dimensional posture models*. They are extracted by applying dimensionality reduction techniques on the dataset P of recorded postures p_i. The resulting low-dimensional space of postures can have arbitrary dimensions d, with $d << 18$. Without loss of generality, in the following explanation, we will use a two-dimensional posture space ($d = 2$).

In Figure 2 (left) we see an example of a low-dimensional posture model. The model was computed based on demonstrations of two-handed grabbing or grasping. Different techniques such as PCA, LLE or Isomap can be used for this purpose. Figure 2 (right) depicts the reprojection error of applying such techniques to the robot postures. We found that for most skills, even with a

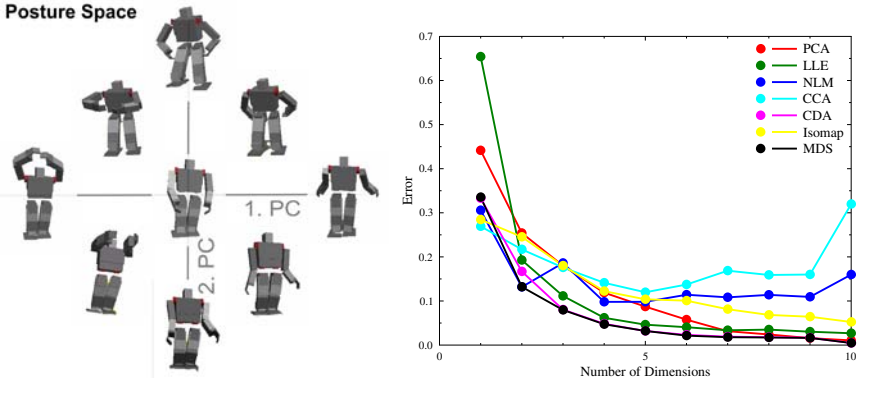

Fig. 2. Left: A representation of the low-dimensional space corresponding to the posture model for grasping with both hands. Right: The projection error resulting from applying different dimensionality reduction techniques to the recorded robot postures.

simple PCA, 95% of the original information can be retained using only four principal components.

Each position in the posture space corresponds to a posture of the robot. In the figure we see the postures resulting from projecting some points back into the space of 18 joint values. Because we get a continuous space, we can compute interpolations and extrapolations of recorded postures. Motions can be synthesized by simply specifying a trajectory in this space. Each point along the trajectory reflects a posture of the robot at a particular time step of the motion.

3.3 Learning

After dimensionality reduction, machine learning techniques are used to optimize the demonstrated skill. As a starting point for the learning algorithm, we use the low-dimensional trajectory of the demonstration. For this, the set P is projected into the posture space, yielding a new set P' of points specifying a d-dimensional trajectory. Next, P' is approximated using n control points $C = \{c_1, .., c_n\}$ specifying a spline curve. The spline can be regarded as a highly compressed representation of the demonstrated motion. Instead of using all points of the original trajectory P' only a limited number n of control points is used for learning.

More precisely, the control points C are used as an initializing individual of a real-coded Genetic Algorithm (GA). A set of slightly perturbed variants of C are created in the initial population of the GA. Each individual is then processed, and the corresponding motion executed by the simulated robot. This is done, by reprojecting each point along the encoded trajectory back to the original space of joint values. Using a user-provided fitness function, each individual is then evaluated and assigned a fitness value. Once all fitness values are determined, the best chromosomes are selected, mated and mutated according to the typical rules of a GA. Finally, when learning is finished, the newly learned skill is applied on the real physical robot.

When performing dynamic motions, such as a standing up behavior, timing plays an important role. Therefore, we add add a special time parameter for each of the control points into the chromosome. The time parameter indicates at which timestep each posture should be realized. Each individual in the GA, thus, consists of the set of values $\{c_1, t_1, .., c_n, t_n\}$.

4 Experiment and Results

To evaluate the proposed approach, we conducted a set of experiment in which a human teacher had to teach a small humanoid robot a set of skills using Kinesthetic Bootstrapping. Among others, the robot learned to perform a headstand, stand up by itself, and walk. In the following we will focus on the standing up and walking skills. In all experiments, PCA was used as a dimensionality reduction technique. The number of dimensions d was set to 4.

The teacher was given about 15 minutes time to kinesthetically demonstrate the respective skill. In Figure 3 we see the result of directly replaying the demonstrated skill. Because of the missing support of the teacher, the robot failed to stand up by itself. Next, we run the optimization as described in section 3.3. The number of control points n was 25. For the 'standing up' skill, the fitness values were determined based on the sum of the z-values (=height) of the robot's head position. The trial was aborted, if the robot's head was below a given threshold, i.e. the robot fell down during the simulation. Figure 4 shows the result of the optimized skill in simulation, and after application on the real robot. As can be seen, the robot learned to stand up by modifying the original motion. In particular, the hip motion was changed such that the robot can lift the torso up, without losing balance (2. picture from right). By moving the hip backwards to an extreme position, the zero moment point of the robot remains between the legs. The result is an elegant solution to the problem of standing up.

For the walking skill, the fitness value of each individual was determined, using the distance traveled from the starting position without falling down. The number n of control points was set to 12. In Figure 5(left) we see the low-dimensional trajectories resulting from the control points of the walking skill before and

Fig. 3. Direct replay of a demonstrated standing up skill by the small humanoid robot. The robot fails to stand up, because of the missing support forces of the human teacher.

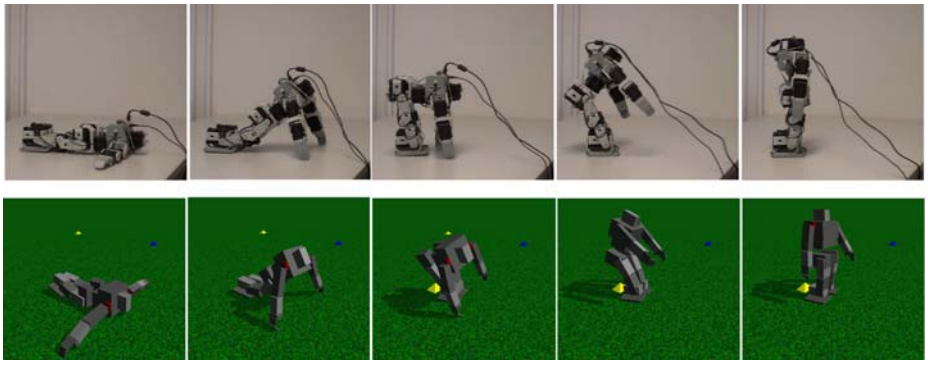

Fig. 4. Results of applying the evolved standing up behavior in simulation and on the real robot. The robot learned to move the hip backwards to an extreme position, so as to pull up the torso without falling forwards.

after optimization. The trajectories show the values of the control points in the first and second principal component. The lower-order components, in this case the first and second component, contain the "most important" aspects of the data. Thus, by visualizing the first two components, we can see most important changes to the robot motion. Figure 5(right) shows the evolution of the fitness values during optimization. With each generation of the GA, the robot managed to travel larger and larger distances. However, after the GA finished, we found that the best individual in the simulation did not lead to a stable walk in reality. This unveils a common pitfall of using a simulator: even the best simulation is only an approximation of the real world. Fortunately, GAs allow for a simple solution to this problem. By testing the best individuals of earlier generations, we can search for solutions that are transferable to the real world. In Figure 6 we see

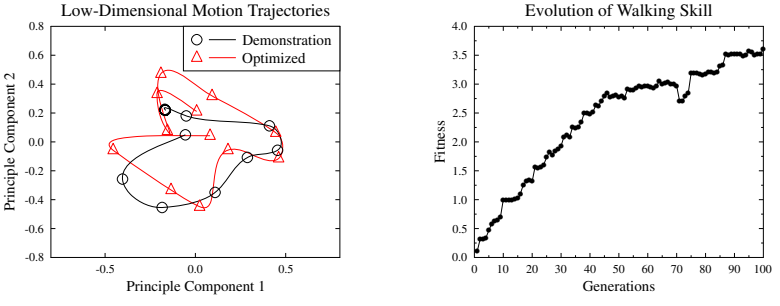

Fig. 5. Left: The trajectories of the walking skill in the low-dimensional space of postures. Right: The evolution of the fitness value during the learning of the skill. The fitness was determined using the distance traveled by the robot.

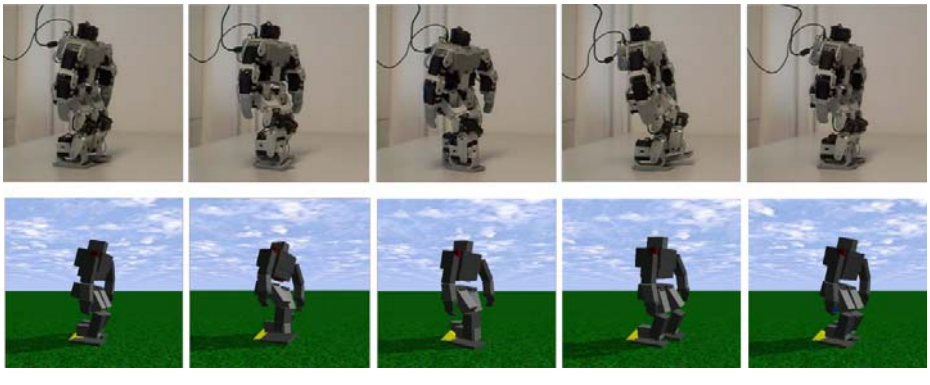

Fig. 6. The result of applying the evolved walking behavior in simulation and on the real robot. Learning this skill only involved a 5 minute kinesthetic demonstration, in which the legs where moved by the human, and the specification of the fitness function.

an evolved stable walking pattern. It corresponds to the fittest individual from generation 50. In later generation, the GA exploited the characteristics of the simulator too much and, thus, generated an individual that was not applicable on the real environment.

5 Conclusion

Up to now, the standard method for creating new motions and behaviors is through low-level programming or through the use of graphical user interfaces. Both approaches are labour intensive and do not support the automatic optimization of the specified behavior. In this paper we presented a new approach to programming humanoid robots, which relies on physical interaction between a human teacher and a learning robot. By optimizing the demonstrated behavior in a virtual environment we can speed up learning times and reduce the need for human intervention. Further, by introducing *low-dimensional posture models*, we were able to integrate human knowledge into the learning process. In the future we will investigate the use of low-dimensional posture models in conjunction with other learning techniques such as Reinforcement Learning or Neural Networks. We also plan to use the described technique on more sophisticated android robots with more degrees of freedom.

References

1. Chalodhorn, R., Grimes, D.B., Grochow, K., Rao, R.P.N.: Learning to walk through imitation. In: IJCAI, pp. 2084–2090 (2007)
2. Nehaniv, C.L., Dautenhahn, K.: The correspondence problem. In: Imitation in animals and artifacts, pp. 41–61. MIT Press, Cambridge (2002)
3. Aleotti, J., Caselli, S., Reggiani, M.: Leveraging on a Virtual Environment for Robot Programming by Demonstration. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems IROS 2003, Workshop on Robot Programming by Demonstration, Las Vegas, USA (2003)
4. Hersch, M., Guenter, F., Calinon, S., Billard, A.: Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Trans. on Robotics (2008)
5. Tani, J., Nishimoto, R., Namikawa, J., Ito, M.: Codevelopmental learning between human and humanoid robot using a dynamic neural-network model. IEEE Transactions on Systems, Man, and Cybernetics, Part B 38(1), 43–59 (2008)
6. McNerney, T.S.: From turtles to tangible programming bricks: explorations in physical language design. Personal Ubiquitous Comput. 8(5), 326–337 (2004)
7. Frei, P., Su, V., Mikhak, B., Ishii, H.: Curlybot: designing a new class of computational toys. In: CHI 2000: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 129–136. ACM, New York (2000)