Towards a Simulator for Imitation Learning with Kinesthetic Bootstrapping

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

Institut für Informatik
Technische Universität Bergakademie Freiberg
E-Mail: bergere@student.tu-freiberg.de, amor@informatik.tu-freiberg.de, vogt3@student.tu-freiberg.de, jung@informatik.tu-freiberg.de

Abstract. This paper presents a physics based simulator that allows kinesthetic interactions between a human and a robot to be recorded, and later used for imitation learning. We argue that kinesthetic interaction can be a very important tool for programming robots, if properly supported by a simulation engine. For this, we propose a new scheme for robot motion learning based on kinesthetic bootstrapping. Interactions with the robot are used to create a low-dimensional posture space. The posture space together with the presented simulation engine allow for fast imitation learning of behaviors. Early results of this approach, using Genetic Algorithms as learning technique, will be presented.

1 Introduction

Robot simulators have been widely used to simplify the programming of robot motion, behaviors and planning. By providing important features, such as faithful physical computations and graphical representations, a simulator can drastically support the user in programming the robot. For example, using such a simulator and Evolutionary Algorithms, we can evaluate many different possible control programs and automatically synthesize new behaviors. The efficiency of this approach has been shown in recent work, such as the research of Karl Sims [5] or Hod Lipson [4]. Still, the approach suffers from low control of the user over the evolved behaviors. Artificial evolution often comes up with innovative but unnatural solutions to posed problems. To avoid this, we present a new kind of simulator which features kinesthetic interactions [1]. The simulator allows interactions between a human and a robot to be recorded and later used for learning purposes. We argue that kinesthetic interaction can be a very important tool for programming robots, if properly supported by a simulation engine. For this, we propose a new scheme for robot motion learning based on kinesthetic bootstrapping. Interactions with the robot are used to create a low-dimensional posture space. The posture space together with the presented simulation engine allow for fast imitation learning of behaviors. Early results of this approach, using Genetic Algorithms as learning technique, will be presented in Section 3.2.

Fig. 1. A human user kinesthetically manipulates a small humanoid robot. The resulting postures are used to bootstrap a learning algorithm.

2 Freiberg Robot Simulator

The Freiberg Robot Simulator (FRS) is a simulation engine which is developed to support kinesthetic user input. Additionally, the simulator should allow to run a particular robot motion both in real or virtual space. The simulator, thus, is able to control either the real robot, or an ODE simulated virtual robot, or both. In all three cases the interface for controlling the robot, which is provided by the server, is the same. This frees the user from having to create different control programs for different platforms. In the following we will explain the architecture of the Simulator in more detail.

2.1 Client/Server

The FRS is based on a Client-Server Architecture. The simulation and the hardware control of the real robot are handled by a server. The server has three different modes in which it can be used. In the *passive* mode, the server reads in each step configuration of the real robot's motors and applies these values on it's simulated counterpart. In our particular case we have a Bioloid humanoid robot and, thus, read 18 motor values and apply them in simulation. In this mode, the user can kinesthetically manipulate the robot. All manipulation and interaction data is stored and can later be used for learning. The second server mode is the *simulation* mode. In this mode, the virtual robot can be controlled via client program. The client sends the desired configuration of the robot's 18 motors to the server and receives information about the environment and the robot. This is done via string messages. Thus, any program that can create a TCP/IP connection and process strings can act as a client. This mode resembles the typical client-server architectures which are also used in other simulators, such as the RoboCup 3D Simulator [3]. In this mode, we can simulate different situations and behaviors of the robot in a virtual environment without having to resort to the real robot. Finally, in the third mode, the *active* mode is identical with the *simulation* mode, with the only difference that now we control the real robot. In this mode, we can apply the learned behaviors on the real robot and evaluate the results.

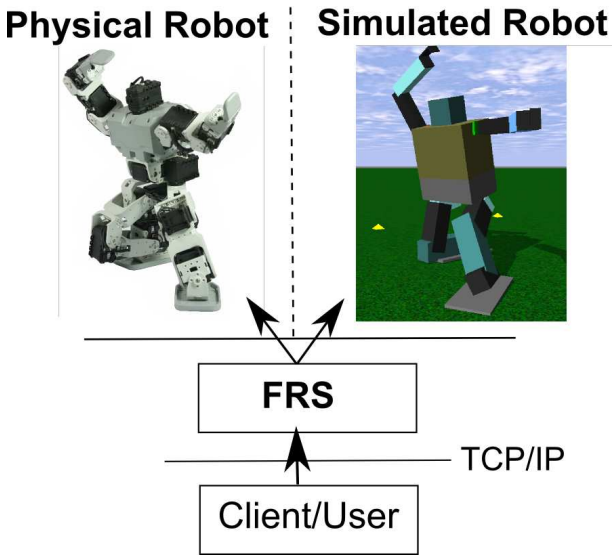

Fig. 2. The Freiberg Robot Simulator: client program communicates via TCP/IP with the server by sending string messages of the desired joint angles. Whether the controlled robot is a real or a virtual robot, is not visible, and irrelevant to the client.

In Figure 2 we see a schematic view of the client-server architecture. As can be seen, it is irrelevant to the client whether currently the real robot or the virtual robot is used. In both cases the communication interface is the same.

3 Imitation Learning

Since the landmark paper of Karl Sims [5], Genetic Algorithms have been used in many studies for evolving artificial creatures and their locomotion behavior. However, the approach still finds limited application in the robotic community. The main reason for this, is the missing user control over the resulting motion. Like real evolution, artificial evolution often comes up with innovative solutions to posed problems. While these can lead to a high fitness function for the provided fitness function, they might not correspond to the users idea of how the desired motion should be. For example, evolution might create a frog-like leaping motion, where the user intended to create a walking behavior. To solve this problem, we use a technique inspired from parenting behavior in humans.

3.1 Kinesthetic Bootstrapping

When learning a new skill such as walking, infants are often supported by their parents. This helps to overcome learning barriers and increases the speed of

learning. While in the pedagogical and psychological literature the broad term for such a process is *scaffolding*. In this paper, we apply the same principle to robot motion learning via Genetic Algorithms. However, we will refer to this as *kinesthetic bootstrapping*, in order to avoid any misunderstandings. It most be noted that, while we explain the principle using Genetic Algorithms, any other learning technique can benefit from the presented approach. Figure 3 shows a diagram of imitation learning with kinesthetic bootstrapping.

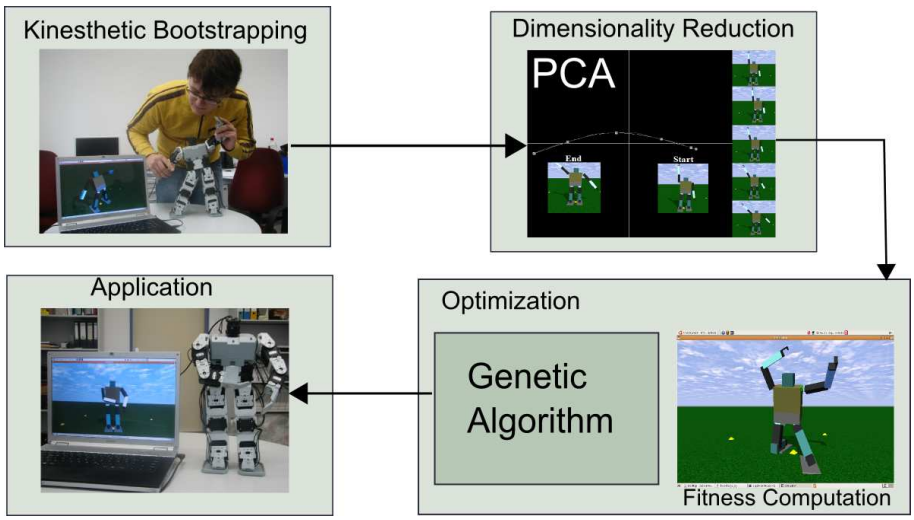

Fig. 3. Kinesthetic Bootstrapping.

First, the user manipulates the robot and moves it according to the desired behavior. For example, for bootstrapping a particular walking motion, the user might move the legs of the robot such that a walking cycle is mimicked. During this, we record the configurations of the robot motor configurations with 20 Hz. This yields a set of 18 dimensional vectors, each representing a particular posture that the robot had during manipulation. Once the user finished manipulating the robot, the recorded data is used to create a low-dimensional space of the 18-dimensional manifold of robot postures. Using Principle Component Analysis (PCA) we create a low-dimensional posture space with dimensionality d with $d << 18$. Throughout this paper we will set the parameter $d = 2$.

The above process yields a 2-dimensional posture space of robot postures. Each 2D point in this space represents a particular posture of our robot. Re-projecting the point back into the original 18-dimensional space allows us to reconstruct the full motor information for the given posture. Further, posture space is continuous, which means that small changes in the coordinates of a point in this space yield small changes in the posture of the robot. Next, the recorded data points are projected on the 2D posture space. This yields a tra-

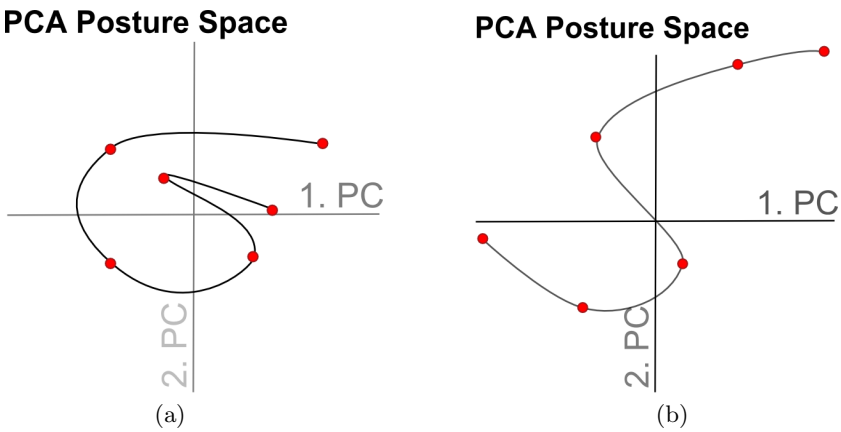

Fig. 4. (a) A stand-up motion trajectory resulting from projection a motion into a 2D posture space. The red dots indicate the sampled control points for the Bezier curve. (b) A modified stand-up trajectory; the control points of the Bezier curve are changed using an learning algorithm (Genetic Algorithm).

jectory as can be seen in Figure 4(a). To further reduce the amount of data used during learning, we sample 6 control points from the projected trajectory (red dots in Figure 4). These are used as a control points of a Bezier curve, which approximates the original motion trajectory. The Bezier curve can be regarded as a highly compressed version of the originally recorded postural data. If we take consecutive points along this Bezier curve and project them back into the 18-dimensional space, we get the postural data for controlling the robot. This means, that we can store a whole motion just by saving the coordinates of the six control points in the posture space, thus 12 values. However, due to the influence of the human user during the manipulation phase, the motion is likely not to produce a stable result. For example, the robot is likely to fall down when reproducing a kinesthetically teached motion for the first time. Similarly to children, the robot needs a trial and error phase in order to adapt the new motion to dynamics. This is achieved using a learning/optimization phase.

3.2 Optimization in a Simulation Environment

In the optimization phase, the kinesthetically bootstrapped motion is refined to achieve a stable and successful motion. We do this by using a Genetic Algorithm (GA). The GA is a *real-coded* and has 12 genes per chromosome. Every pair of genes codes the XY coordinates of a Bezier control point in posture space. Thus, each chromosome corresponds to one motion trajectory. The chromosomes are initialized as randomly perturbed (mutated) versions of the original motion trajectory. Then, each chromosome is evaluated by performing the corresponding motion in the *simulation* mode of the server, and a fitness value is assigned. The

fitness function is dependent on the task to solved, and can be, for instance, the distance traveled by the robot. Once the fitness is evaluated the best chromosomes are selected, mated and mutated according to the typical rules of a GA. Finally, when learning is finished, the server mode is changed to *active* and the motion trajectory is applied on the real robot.

The difference between this approach and previous research on evolution of motion control, is the of the posture space. By kinesthetically manipulating the robot the user provides important data that is used to bias the genetic algorithm to particular poses. Irrelevant or undesired robot poses are discarded from the search process. Thus, the user can exert much more control over the search process of the GA than in previous research. Kinesthetic bootstrapping is easy for humans to understand and perform. Therefore, it is a far easier than trying to change the fitness function or other parts of the GA. Additionally, it has the advantage of speeding up the learning speed, because the number of parameters is reduced from several hundred (18 $\times$ time-steps) to only 12 values.

Fig. 5. Original stand-up motion replayed by the robot after kinesthetic bootstrapping. The robot does not rise too high up.

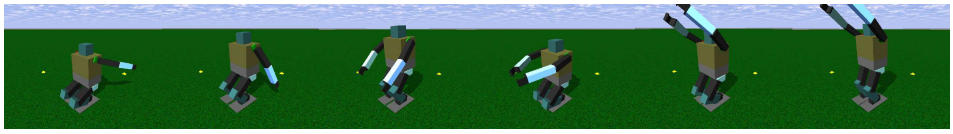

Fig. 6. Optimized stand-up motion. The robot rises higher, because of the new swing of the arms.

3.3 Early Results

While all components of our learning mechanisms are working and already show promising results, we are still in the process of performing quantative evaluations and making first experiments on complex learning tasks. The following example shows early results of performing imitation learning using FRS and kinesthetic bootstrapping. During the kinesthetic bootstrapping stage we manipulated the robot such that it stood up, and raised, and bent its arms. This motion can be seen in Figure 5; the pictures correspond to the Bezier control points of Figure 4(a). Next, the motion was mutated and changed as described in section 3.2.

The result is the stand up motion in Figure 6 and Figure 4(b). We can see that the GA changed the motion such that the robot can rise higher, by replacing the positions of the Bezier control points in the posture space. Still, the style of the stand-up motion as the user intended it to be is retained by the GA.

4 Conclusions and Future Work

This paper demonstrated that kinesthetic interaction can be an important tool if properly supported by robot simulators. It allows to program robots in a more natural way. Most importantly, we showed that it also helps to improve quality and speed of imitation learning. For this we proposed an preprocessing step to learning, which we call *kinesthetic bootstrapping*. The approach is based on psychological findings about parental behavior. Additionally, we introduced the Freiberg Robot Simulator and showed how control programs can be written such that they can be evaluated on real and virtual robots without any special-purpose code. This and the support for kinesthetic interactions not only improved the usability of the simulator, but also allowed for a scientifically interesting new approach to imitation learning. For future work we hope to conclude our experiments on kinesthetic bootstrapping and imitation learning. Our particular interest is the Action Capture [2] technique, which is an imitation learning method which records interactions with objects in virtual environments. Further, we hope to solve some difficult learning tasks such as robot soccer skills with our approach. Further, we want to investigate how much speed can be gained by this approach over traditional learning approaches.

References

1. M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *IEEE Transactions on Robotics*, 2008. Accepted.
2. B. Jung, H. B. Amor, G. Heumer, and M. Weber. From motion capture to action capture: A review of imitation learning techniques and their application to VR-based character animation. In *Proceedings VRST 2006 - Thirteenth ACM Symposium on Virtual Reality Software and Technology*, pages 145–154, 2006.
3. O. Obst and M. Rollmann. SPARK – A Generic Simulator for Physical Multiagent Simulations. *Computer Systems Science and Engineering*, 20(5):347–356, Sept. 2005.
4. J. B. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby, and R. A. Watson. Evolutionary techniques in physical robotics. *Creative evolutionary systems*, pages 511–523, 2002.
5. K. Sims. Evolving virtual creatures. In *Computer Graphics, Annual Conference Series, (SIGGRAPH 1994 Proceedings)*, pages 15–22, July 1994.