Learning by demonstration using physical manipulation of a robot model

Rory Breuk [a]

[a] *Plantage Muidergracht 22-24, 1018TV Amsterdam*

Abstract

In this paper we introduce a way to teach robots new actions by manipulating a model of the robot. We will create a communication level between the model and a Sony AIBO. By using simple learning algorihms an AIBO has to mimic the movements a human operators demonstrates by playing around with the model.

Contents

0.1 Introduction

In human-robot interaction (HRI) many different disciplines are studied. One of these is how robots can learn new actions. When robots have the ability to learn we can expand the set of a robot's actions without the need of complicated programming or other ways in which very precise specificatons of the pose of robots have to be defined.

A lot of different ways of learning have been studied over many years. Inductive supervised learning is a research field in which machines autonomously try to optimize a performance function using a dataset and prior knowledge about its nature. The problem with these methods is that sometimes a dataset is not available and the performance function can be hard to define. Especially when the performance differs by the long-term effects of an action or certain previous or next actions greatly influence the performance. [3]

Learning without a dataset can also be used to learn new actions. The robot has to try all possibilities to perform the action in order to test the performance. This method however, is rarely applied in the real world, because trying the possibilities may result in problems when performing them and it can be very time consuming to execute all of the possible ways the action may be performed.

A last way to learn a new action is to learn from the way humans perform it. This method is called learning from demonstration. In learning by demonstration the robot by some way observes how humans would do the action and try to find a way to optimally mimic it.

The aim of this paper is to test how well learning by demonstration performs when a model of a robot is used to demonstrate the actions. For this will create a model of the Sony AIBO ERS-7 robot and use it to demonstrate actions to be performed by the robot. We will use the model to record actions and optimally execute them on the AIBO.

We will test the system by performing a task where the AIBO tries to sit down from a lying posistion.

0.2 Theory

In this project, we consider an action as a certain amount of transitions between different configurations of joint positions. These configurations are different states of the robot. We need to read data of the use of the model of the AIBO and make states of them. Then we want to learn which state transitions we want to use in the action.

We had to consider the fact that it is unwanted to demonstrate an action many times. In most learning methods we often use datasets of hundreds or even thousands of examples, we only want to use between the 5 and 10 demonstrations. It should even be possible to learn an action in only one demonstration. For this reason we will not use very sophisticated learning methods, but rather focus on proper generalizations during the action.

We also did not want the movements of the different joints to influence eachother too much. Points in time where one joint has an important state, the states of other joints may not be important. This is why we learned the states of the joints independently.

0.2.1 Sampling

As a rule of thumb we considered the places where the joints are barely moved (the extremes) as key states of the joints. Places in which the joints are moved are considered as transitions. To find these places, we had to find where the derivative of the movement is (close to) 0.

To find these places we convolved the function with the derivative of the Gauss function:[2]

$$\frac{\delta G\sigma}{\delta x} = -\frac{x}{\sigma^2}G \tag{1}$$

In which G is the Gaussian at scale σ and σ the scale of the derivative. In the implementation the σ is a variable which the user can change.

Because we used the Gaussian the results of the convolution are smoothed. This is a desirable side-effect, because when one is using the model a lot of hesitations will occur. People are not able to make movements as smooth as robots, even while it is desirable.

We then made a timeline which indicates the states the begining and the end of the state in which the joint isn't moved. Because it is impossible to hold a joint entirely still we used a theshold which indicates which values close to 0 can be considered as no movement.

We use a value S to indicate the amount of states we want to define. Starting we rise the threshold, causing the states to decrease. When the amount of states is equal to S we are ready with sampling a demonstration.

Because every demonstration is different we have to determine the threshold on every demonstration and for every joint.

0.2.2 Generalization

Having a timeline for every joint, all with the same amount of states in the timeline, makes it easy to generalize between all demonstrations. The events on all timelines of a joint are averaged between the demonstrations and the angle of the joints in the states are also averaged.

Finally the timelines of the final action is created. One run in which transitions are executed by the AIBO using the `Pyro` framework (which will be discussed later) is about 0.3 seconds. We try to fit the timelines to the frames by moving the time to the frame closest to the event.

0.2.3 Calibration

The sensors in the model have to be calibrated. Because we used linear sensors and the AIBO joints are also linear we only need to know the start and the end values of the degree of freedom of the joints the model allows. The values of the joints on the AIBO in `Pyro` are known to be between -1 and 1. The value of the sensor is converted to the value on the AIBO by:

$$v_{aibo} = 2 * \frac{v_{model}}{\max(v_{model}) - \min(v_{model,max})} - \min(v_{model}) \tag{2}$$

Using this equation we can use the learned states of the model to be executed by the AIBO.

0.3 Application

0.3.1 Model

We made a wooden model of the AIBO on which the sensors can be attached. The most important parts on actions of the AIBO are usually performed by the legs. Therefore we only modeled the legs of the AIBO.

The joints on the legs on the AIBO are defined by:

Rotator The shoulder joint connected to the body of the AIBO

Elevator The second shoulder joint connected to the rotator, used to elevate the leg

Knee The knee of the leg

The model had to somewhat have the same physical constraints as the constraints on the AIBO. Since the model has to be calibrated, these do not have to be exactly the same, but the use of the model is more natural when they do more or less match. The resulting model is shown in image is shown in figure 1 and figure 2.

0.3.2 Circuit

We used the open source prototyping platform Arduino Duemilanove for reading sensor information. The arduino provides on-board serial USB communication which makes communication with the computer very easy. The board also has 3.3v and 5v output ports, 6 analog input ports and 13 digital outputs. The Arduino has the opportunity to make shields for it, so circuits can be easily attached and detached to it.

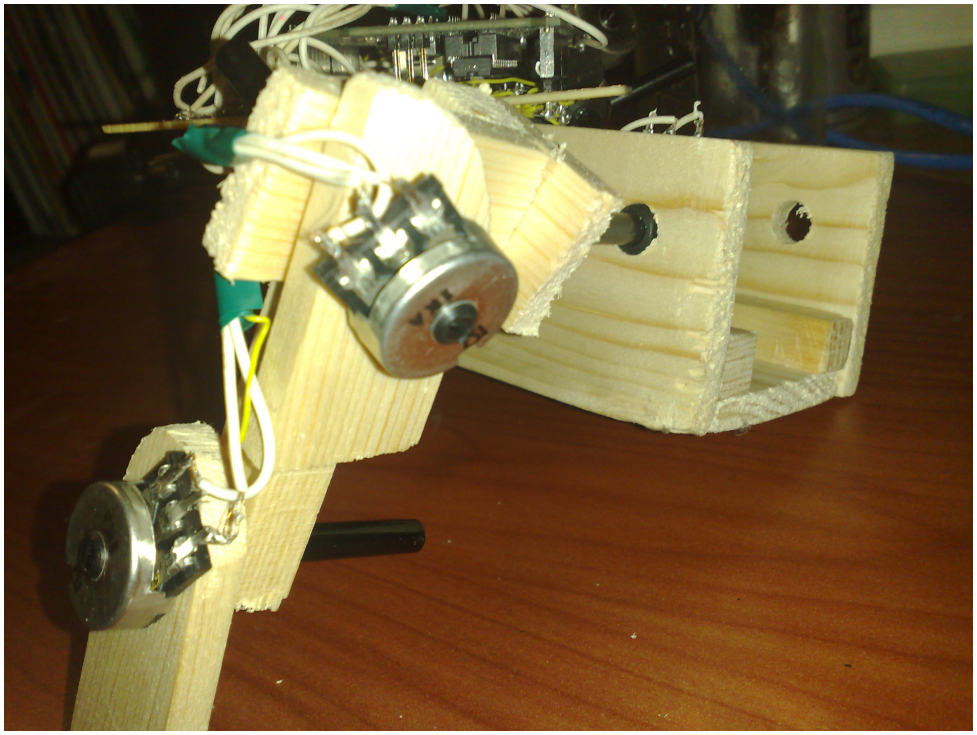

Figure 1: One leg of the model

The turn angle of the joints on the AIBO are physically limited. For this reason we were able to use a linear single turn potentiometer to measure the turn angle. We could also use rotary encoders, but with the use of a potentiometer we wouldn't need a clock so clock syncing (which may cause problems) would not be needed.

Since we want to model the pose of all four four legs of the Sony AIBO, which all have three joints, we needed to examine a total of twelve joints. The Arduino only has 6 analog input ports, so we needed to multiplex between the potentiometers. We used the 4051 multiplexer for this task.[1] The schematic is shown in figure 3 and the circuit is shown in figure: .

We also connected a dimension 3 Axis 3g analog accelerometer on the shield. We used this so the exact pose of the model can be determined. For example, when the model is held sideways or upside-down we could use the accelerometer to find this pose. For this project we did not use the data of the accelerometer, but for future purposes of the model it might be usefull. Because this part of the shield wasn't used in our experiments this isn't shown in the schematic.

0.3.3 Learning

We used a straightforward implementation of the sampling and generalization discussed above. The user first has to calibrate the model. The program asks the user to fully turn the joints to its maximum and minimum value. A user has to perform the action on the model a small amount of times and the algorithms create a series of frames to be executed by the AIBO. The learned values are written to a file.

0.3.4 Execution on Sony AIBO

We used the Pyro framework for controlling the Aibo wirelessly by the computer. The project is intended to provide a python-based programming environment for many different types of robot. One of these robots is the AIBO.

The framework uses a so called `Brain` object to define the poses of joints in frames. We made a brain which reads the frames and converts them to values of the AIBO.

Figure 2: One leg of the model

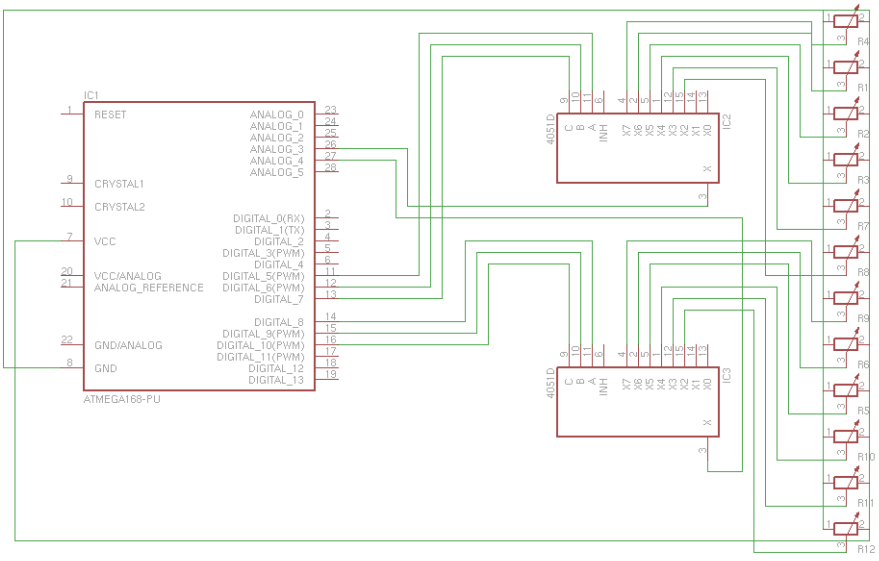

Figure 3: The schematic of the Arduino shield

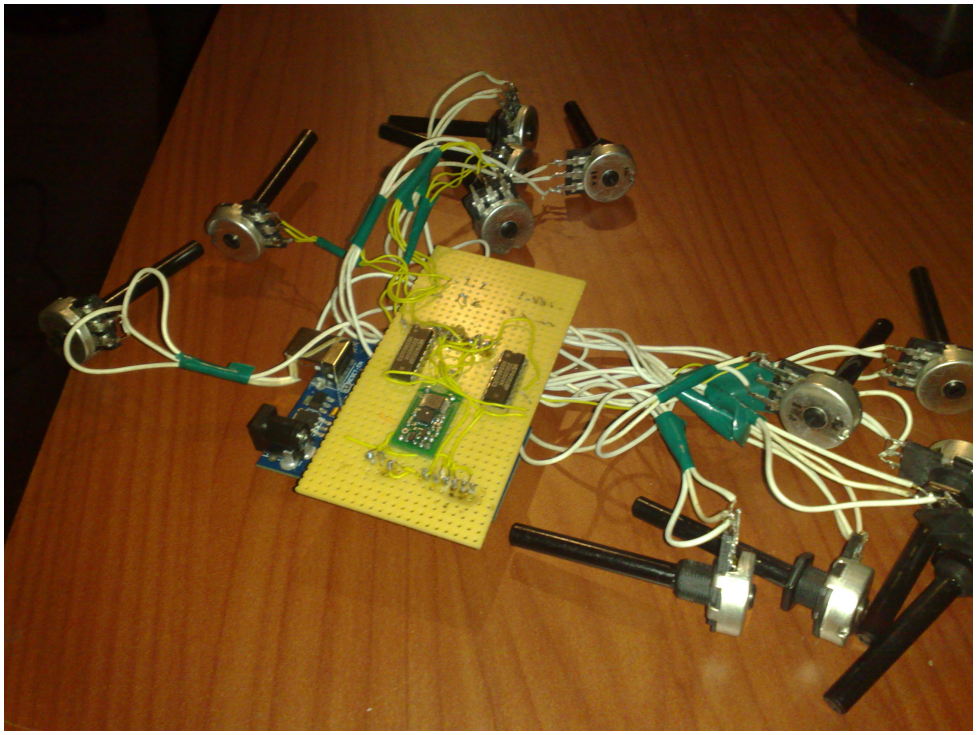

Figure 4: One leg of the model

0.4 Experiments

We tested the program by having an AIBO lying down move to the sit position. It is not possible to exactly measure how good an action is performed, so we have to make a description on the performance of the system. We used different values for the amount of states per joint S and the scale of the derivative of the movement σ. For the experiments we used 8 demonstrations.

0.4.1 Results

S	$sigma$	What happens
2	0.2	The execution fails entirely
5	0.2	The AIBO almost sits down, but falls in the end
8	0.2	The AIBO sits down, the movements is not very smooth
10	0.2	The AIBO sits down, the movements are even more hesitating
8	1	The AIBO sits down and the movements are pretty smooth
10	1	The difference is not very big
8	5	The AIBO barely sits down
10	5	The AIBO does sit down, but the performance is less than with a smaller σ

0.5 Conclusion

The aim of this paper was to find a better way to teach actions to a robot. The system performs pretty well even though we didn't use a lot of demonstrations. When only a very small amount of frames are used the performance is very bad, but when the frames are increased the performance will also greatly increase. However, the smoothness of the system will decrease. This is probably because the hesitations of a user have more influence on the execution.

0.6 Discussion

We used very simple algorithms in order to learn the movements. One of the important parts of the paper was the creation of the model, so we spent more time in improving the model and making a communication framework between the computer and the AIBO. This framework is, however very easy to adapt so it is easy to improve the learning algorithms and so the performance of the AIBO. The system may also be easily adapted for direct manipulation, making it possible for users to directly execute certain movements on an AIBO by moving the joints on the model.

Bibliography

[1] Tomek Ness. Analog multiplexer/demultiplexer 4051, 2006. http://www.arduino.cc/playground/Learning/4051.

[2] L. Dorst R. v/d Boomgaard. *Computer Vision*. 2007.

[3] S. Schaal. Learning from demonstration. *Advances in Neural Information Processing Systems*, 9:1040 –1046, 1997.