Learning a Table Soccer Robot a New Action Sequence by Observing and Imitating

Dapeng Zhang and **Bernhard Nebel**
Research Group on the Foundations of Artificial Intelligence
University of Freiburg
Germany D-79110

Abstract

Star-Kick is a commercially available and fully automatic table soccer (foosball) robot, which plays table soccer games against human players on a competitive level. One of our research goals is to learn this table soccer robot skillful actions similar to a human player based on a moderate number of trials. Two independent learning algorithms are employed for learning a new *lock* and *slide-kick* action sequence by observing the performed actions and imitating the relative actions of a human player. The experiments with Star-Kick show that an effective action sequence can be learned in approximately 20 trials.

Introduction

Table soccer (or foosball) is a popular table game in which two teams play against each other. Each of them controls four game rods, which are called attacker, midfield, defender, and goalkeeper. The playing surface is $1200mm$ in length and $680mm$ in width. There is a goal at both ends of the playing surfaces. Each team tries to score goals while defending against attacks from the opponent.

Star-Kick is a robot that plays table soccer games with human players. Under the playing surface, a black-white camera observes the ball and eight DC motors control the movements of the four game rods. Each motor can either move or turn a game rod. A processing cycle for the whole system takes about $20ms$, which is already fast enough to play against human players on a competitive level. Statistics show that our robot wins approximately 85% of all games (Weigel 2005).

So far, Star-Kick shows a good performance on actions such as *kick*, *block*, and *clear*. Human players are able to do more complex and interesting actions such as *lock*, *dribble*, and *slide-kick*. A *lock* action locks the ball moving along the direction perpendicular to the game rods. A *slide-kick* action passes the ball along the rod direction and then kicks it towards the goal. Performing these actions would dramatically increase the ability of Star-Kick. However, using uniform parameters with different playing figures to successfully perform actions such as *lock* is impossible due to the current system accuracy.

The system accuracy of Star-Kick is affected by the accuracy of the sensors as well as the accuracy of different mechanical parts and the way in which these parts work together. Assuming that the environment is ideal, the calibration process of Star-Kick establishes a coordinate system which brings the ball and all the rods together in a playing field. This assumption entails that the vertical axes of the playing figures mounted on the same rod are in a plane and that the camera image restored from the projection and distortion is linear with respect to the coordinate system. In reality, however, these assumptions do not hold, and there are enough correlated errors to affect the successful execution of actions significantly. Although many system errors are inevitable, Star-Kick does well in repeating the same performance in the same situation. This characteristic can also be found in other systems, and is generally called *repeatability*. Typically, the repeatability of a system is much better than its accuracy.

Learning algorithms often require thousands of trials to converge to an optimal behavior. With Star-Kick, however, a learning process that requires one thousand trials would be too much for the final users who want to enjoy the games. Learning by imitation is one of the methods which can dramatically reduce the number of required trials. It utilizes the experiences of others as a shortcut to good outcomes. It is particularly interesting to imitate human actions with Star-Kick because playing in a human-like fashion makes the game more interesting for human opponents. Furthermore, it is possible to interpret the human actions of moving and turning the rods in terms of motor commands, making it easier to imitate them.

Assuming that Star-Kick can repeat an action with the same set of parameters, two independent learning methods are implemented to find a proper set of parameters for the action sequence *lock* and *slide-kick*. One method searches the parameter space to implement the action of locking a moving ball. The other method is used for finding a set of motor commands to perform a *slide-kick* action similar to the abilities of a human player. With these methods running parallel, the experiments show that Star-Kick acquires an effective sequence of *lock* and *slide-kick* after about 20 trials.

Related Work

The first table soccer robot KiRo, which is the prototype of Star-Kick, was developed at the University of Freiburg (Weigel & Nebel 2002). The actions of both KiRo and Star-Kick have been developed by hand-coding, whereby also sophisticated actions such as *lock* and *slide-kick* have been implemented (Zhang 2005). Unfortunately, these are not adaptive to different playing figures and therefore cannot be used in a market-available product such as Star-Kick.

Recently, table soccer robots have become the research topics in at least five universities. However, there do not seem to exist any formal publications from the others to date. Nevertheless, a Wikipedia page shows that researchers from Rice University also developed a table soccer robot called *foosbot* which plays table soccer games with human players on a competitive level (foosbot 2006). A video of *foosbot* shows that it can do some actions similar to *slide-kick* and is controlled by some "control laws" which are described in the Wikipedia page.

Learning by imitation or programming by demonstration has been an important branch of artificial intelligence for years. It is particularly interesting in connection with humanoid robots, which can be difficult to control because of their high degrees of freedom (Schaal 1999; Billard & Siegwart 2004). It can also be found within other systems for example a mobile robot (Smart & Kaelbling 2002) and a virtual intelligent agent (Price 2003). The essential point of programming by demonstration is to reduce the learning costs by observing the performance of other systems, which is very useful for our table soccer robots. The typical steps for programming by demonstration, such as observation, interpretation, and replication (Dillmann *et al.* 1999; Mataric 2000), can also be found in our work.

Similar to learning by imitation, behavioral cloning (Sammut *et al.* 1992) takes the control-commands of human operators as input. It is different from our work in several aspects. Behavioral cloning induces rules from the command sets of a number of human operators, while the learning algorithms in our work improve the action sequence over the trials by imitating only one input set. In addition, behavioral cloning finds some general rules to achieve a complex task. In the imitation process our work aims at mimicing a single human action as closely as possible.

Our ideas in this paper are similar to the ones used with a humanoid robot to play air-hockey (Bentivegna *et al.* 2004). The two systems have similar problems, such as system errors, which effect the performance of the actions, also the learning process cannot be too long etc. Additionally, air-hockey and table soccer are both games with a target running in a plane and two sides playing against each other. Consequently, the solutions are similar from the aspect that the learning is structured, the feedbacks of the performed actions are analyzed to retrieve the information, and the robots learn to play the games from the human players. From a different perspective, the humanoid robot is very different from Star-Kick. Thus the mathematics models used to control the humanoid robot, such as "locally weighted projection regression" are not a proper solution for Star-Kick.

The Learning Problems

In the following, we describe two independent learning problems. The first one is concerned with learning the action *lock*. The second problem is about learning the action *slide-kick*. We call the direction perpendicular to the game rods as x direction, while the direction along the rod is y direction.

Action *Lock*

In order to lock a ball running in the x direction the active playing figure waits at an angle a_w and position y_w. Then, the ball's x position in the near future $x_{t+\delta t}$ is predicted by its current speed and its current position x_t. Finally, the playing figure is turned to lock the moving ball when the prediction $x_{t+\delta t}$ arrives at a waiting position x_w. Figure 1 shows the sequence of a *lock* action and the involved parameters.

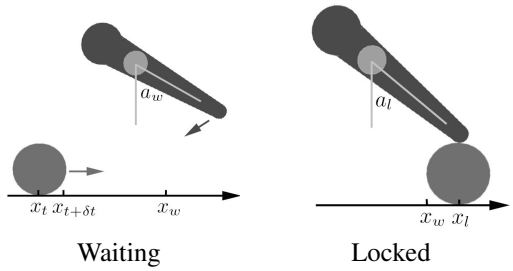
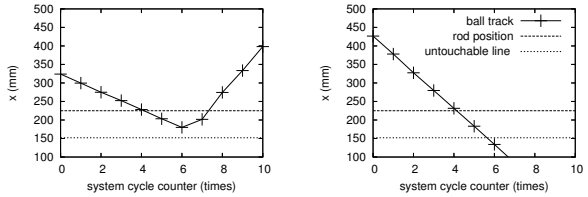


Figure 1: Parameters of a *lock* action

As y_w is kept concurrent with the ball's y position, which can be easily obtained in the games, and $x_{t+\delta t}$ is given by the ball model, we only focus on the unknown variables a_w and x_w. Ideally, these variables are given by the calibration and would be uniform for every playing figure. However, the hand-measured values for all playing figures in a Star-Kick turn out to be $a_w \in [0.74, 1.05(rad.)]$ and $x_w \in [29, 61(mm)]$.

If the parameters are detrimental to lead a successful action, we consider two situtations that the ball is bounced back by the playing figure, or it is not touched at all. Although there can be more reasons for a failed *lock* action, we will focus on these two situations since they are helpful for the parameters' adjustment. Figure 2 shows these situations, where the horizontal in the graphs is the counter of the system cycles, or simply time; the vertical is the x coordinates of the game ball. The area between the upper and lower line in Figure 2 shows the range where the ball can be touched by the playing figure. We say a lock action is *too fast* if the estimated parameters (a_w, x_w) are lower than the expected values which would lead to a successful *lock* action, and it is *untouched* if the estimated parameters are higher than the expected values. Since the time point t_t is known when the playing figure is turned, the ball's track near t_t can be recorded and easily classified into *too fast* or *untouched* by considering the rod's position and its untouchable line. Thus, the feedbacks of a failed *lock* action are known.



Too Fast Untouched

Figure 2: The feedbacks of a failed *lock* action

If the performed actions successfully lock the ball the outcome of the parameters (a_w, x_w) is positive. Additionally, from the stopped playing figure and the locked ball, we know an angle a_l and an x distance x_l, which are shown in Figure 1. Three situations are illustrated in Figure 3, in which the game ball is "close locked", "optimally locked" or "far locked". In the table soccer games, the "optimally locked" position always offers more chances to continue successfully with other actions such as *slide-kick*. Human players have the possibility to dribble the ball to the "optimally locked" position if the ball is "close" or "far" locked. In the figures, the "optimally locked" point has the highest absolute a_l, where the angle of a playing figure is 0 if it is upright.

Close Locked Optimally Locked Far Locked

Figure 3: the situations of a locked ball

At the very beginning of the learning process, the parameter set is initialized to some guessed values $(a_{w(0)}, x_{w(0)})$, which are calculated from the length and thickness of the figure, the diameter of the game ball, and the theoretical position of the rod. The parameters are updated according to equation 1 every time the feedbacks are available. In the equation, d_{ai} and d_{xi} are the learning discount factors. f_{ai} and f_{xi} denote the feedback values of the last performed action.

$$\begin{cases} a_{w(i+1)} = a_{w(i)} + d_{ai} * f_{ai} \\ x_{w(i+1)} = x_{w(i)} + d_{xi} * f_{xi} \end{cases} \quad (1)$$

To simplify the problems, a little update unit s_a is defined for the parameter a_w, and s_x for the parameter x_w. An update only changes the parameters by adding or subtracting one or more update units. As a_w and x_w cannot be unlimited values, the searching space is reduced to a set of discrete values for the parameter set (a_w, x_w).

In order to choose a proper d_{ai} for an update, the number of the successfully performed *lock* actions n_l are recorded for each parameter set (a_w, x_w). The higher n_l is, the lower is d_{ai}. If the ball has never been successfully locked with a parameter set, d_{ai} is set to 1 for the set, which means the feedback is not discounted.

In case the ball is not successfully locked by the performed action with the parameter set $(a_{w(i)}, x_{w(i)})$, f_{ai} is set to -1 if the trajectory of the ball can be classified as *untouched*, and 1 for the classification of *too fast*. In case the ball is locked by the performed action, angle a_l and position x_l can be obtained from the performed action. a_l is used to estimate the quality of the action because the "optimally locked" point should have the highest absolute a_l. If a_l has the highest value among all locked cases so far, f_{ai} is set to 0, which leads to an untouched "updated" angle. Otherwise, the feedback f_{ai} is calculated from the difference between the locked distance x_l and the x_l which leads to the maximum a_l. Also, we have chosen a factor to map f_{ai} into $[-0.5, 0.5]$ for a proper feedback.

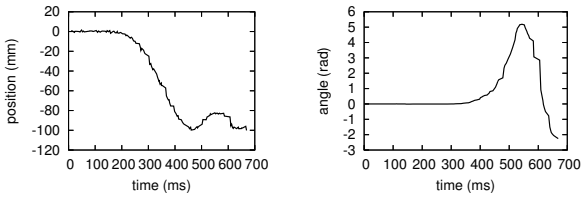
In short, the learning algorithm will continue adjusting the lock waiting angle a_w with an angle step s_a when the ball has never been locked. Afterwards, the adjusted angle is discounted by the number of the successful trials and the quality of the actions.

Different from d_{ai}, d_{xi} is set to -1 or 1 in the following two situations, otherwise it is set to 0. One situation is that the waiting angle $a_{w(i)}$ goes beyond the predefined angle limitations. The other is if a new estimated "optimally locked" position is found, d_{xi} is chosen to move $x_{w(i+1)}$ towards x_l. f_{xi} is set to s_x in both situations. Theoretically, x_w would equal x_l. However, as the sensor data is noisy, sometimes x_l is only an error. The adjustments of a_w offer some chances to explore the area near x_l. If better positions could be continuously found, x_w will eventually arrive at x_l. Otherwise, it only moves towards x_l with a small step s_x.

Observing these two learning updates together, we can see that if a new estimated "optimally locked" position is found, the angle will retain the last value. Because x_w is adjusted by a small value, the parameter set as a whole is not changed dramatically. Hopefully, the *lock* action following the update will still be successful.

Action *Slid-Kick*

Slide-kick is one of the possible actions following the action *lock*. Figure 4 shows the movements of the game rod during a *slide-kick* action of a human player, where the horizontal in the figure is time. The values of the position and angle are "normalized" by setting their start points to zero, so that the actions of Star-Kick can be compared with the human action even if the start points of the actions are very different. From the "angle sequence" and "position sequence" we can observe that the action mainly passes the locked ball along the game rod and then kicks it.



Angle sequence Position sequence

Figure 4: The action *slide-kick* by a human player

Taking the information in the above figures as input, Star-Kick needs to translate these data into its own motor commands. An important principle in our learning method is to use as few motor commands as possible. The first reason is that the feedback data of Star-Kick is not as informative as the data of the human action. For example, there are 146 data points that describe the action of a human, but only 17 data points from Star-Kick. The second reason is that sending too many motor commands within a short time span always causes performance problems with Star-Kick. If a motor needs to be turned with a speed s_t from a standstill, it will need some time δt to accelerate to the required speed. If another motor command is sent before δt, the motor will always be in an accelerating condition. And the third reason is "the simpler, the better".

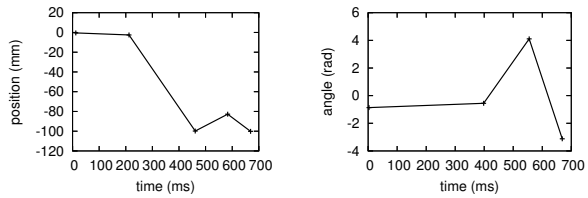


Figure 5: The filtered data of the *slide-kick* action

In order to describe the human action by a few motor commands we need to find out the "significant" movements while ignoring the "irrelevant" ones. Therefore, a filter is implemented to achieve this task. Figure 5 shows the filtered data, in which every line segment with a significant slope k_{hi} and its start point s_{hi} is translated as a motor command. Thus, the action is expressed by a series of (m_i, t_i) where m_i is the speed of the motor and t_i is the time to send the command. As (m_i, t_i) is a basic element which cannot be divided, it is also called "motor primitive".

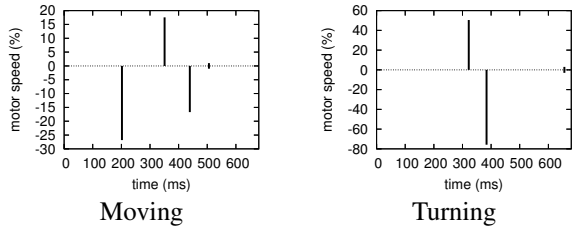


Figure 6: Motor primitives for imitating *slide-kick*

We illustrate an example of motor primitives in Figure 6. Now, the slope of a line segment in Figure 5 could be mapped with a linear function to (m_i, t_i) in figure 6. This solution, however, assumes that the motor will achieve a speed m_i after a constant delay δt. In reality, δt depends heavily on the current status of the motor. Both m_i and t_i need to be adjusted according to the situations.

Figure 7 shows an example of a performed *slide-kick*. Using these data as the feedbacks, and assuming that the turning and moving are two independent actions, a learning algorithm is implemented to refine the motor commands. The refinement is shown in equation 2.

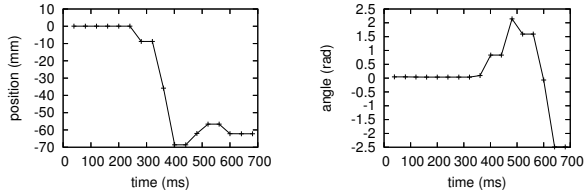


Figure 7: A *slide-kick* performed by Star-Kick

$$\begin{cases} m_i(t+1) = m_i(t) + w_1 * (k_{hi} - k_i(t)) \\ t_i(t+1) = t_i(t) + w_2 * (s_{hi} - s_i(t)) \end{cases} \quad (2)$$

In the equation, t is the trial index and k_i is the slope of the line segments which are obtained by filtering the feedbacks. s_i is the start point of the line segment. w_1 and w_2 are small constants manually chosen as learning rates.

The overall similarity between the original human action and the imitation of that action is also taken into consideration. As a measurement for the similarity, the difference of the actions is calculated in the following way.

$$d_a = \sum_{i=1}^{n} \sqrt{(v_i - v_{hi})^2} \quad (3)$$

In the equation, n is the number of the feedbacks. v_i is the angle or position values of the performed action, v_{hi} is the relative values of the original human action. The two actions are more similar if d_a is smaller.

The learning algorithm performs a fixed number of trials using the refinements defined in equation 2. Then the amount of the refinements is discounted with the help of the overall similarity d_a. In other words, the learning algorithm helps to search a sub-optimal overall similarity. If the refinements do not help to improve the similarity, the learning algorithm will continue to try some motor commands closer to the known "best" case.

Experiments

The experiments described in the following were conducted with a Star-Kick. The scenario of the experiments is shown in figure 8. The game ball is passed manually from the midfield. The passings are with intentional variations of speed, as in real games of human players. Then, the *locked* action is performed by the middle playing figure of Star-Kick's attacker. If the ball is successfully locked, Star-Kick will continue with a *slide-kick* action. The trajectory of the ball follows the arrows roughly. If the first *lock* action fails, Star-Kick is reset to do the *lock* action again. In the scenario, the forward direction of the ball is blocked by the opponent's playing figures, which means a direct *kick* will never bring the ball into the goal.

Over 100 trials are carried-out within the defined scenario. Figure 9 shows these results. If the game ball is successfully kicked into the goal with the action sequence, a reward with value 1 is gained. If the game ball is successfully kicked but does not roll into the goal, a reward with value 0.5 is gained.

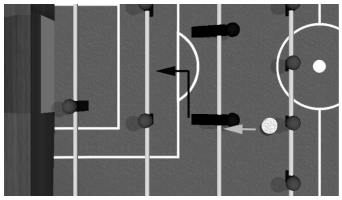


Figure 8: The scenario of the experiment

Otherwise the reward value is 0. The results are smoothed by averaging the rewards over 9 trials. In the figure, the horizontal is the number of performed trials; the vertical is the average rewards for each trial.

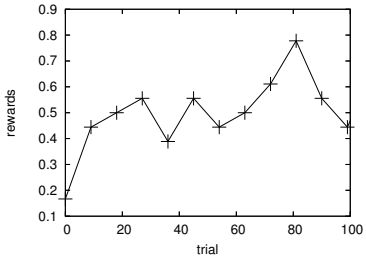


Figure 9: Learning Curve for the action sequence

We can see from the results that the action sequence is improved during the learning since there is an increase in rewards from 0.16 to 0.5 in about 20 trials. In the very beginning, the action sequence can hardly achieve any rewards. After the second data point, the 18^{th} trail, the rewards oscillate around 0.5. Since a successful *lock* action actually triggers the learning process of the next action in the action sequence, *slide-kick* is not performed in every trial in this learning curve.

Figure 10 shows the rate of the successful *lock* actions during the same learning process as in Figure 9. The game ball can be locked with a probability of approximately 80% after Star-Kick performed more than 20 trials with the help of the learning, while the trials at the beginning have a success of approximately 55%. We need to mention here that the very first trial of the *lock* action is hardly successful due to the guessed parameters. Also the learning curves would be much better if we alternated the evaluation and the learning. The evaluation is done in this way because we want to show that the learning of the action sequence can be achieved while human players actually play Star-Kick.

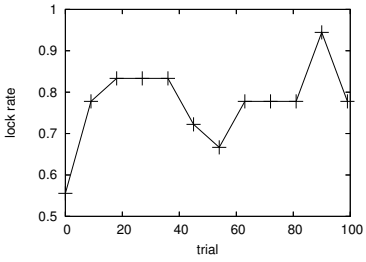
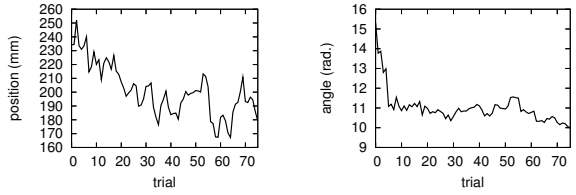


Figure 10: lock rate

There are mainly two reasons for the fluctuation in the learning curves of figures 9 and 10. The first reason is that the games are not fully observable and predictable. Errors from other parts of the system, for example the ball model, are inevitable. The second reason comes from our assumptions that Star-Kick can repeat the same performance in the same situation, and that the turning and moving are independent in the imitation. The experiments show that with these assumptions the performance is better, but they are not dependable to the extent that they are 100% right.



Position Difference Angle Difference

Figure 11: Difference between human and robot actions

With the improvements of the overall performance, not only the *lock* action, but also the *slide-kick* action is improved during the learning process. Figure 11 shows that the values of the angle and position differences defined in equation 3 are decreased during the learning. The values are averaged over 10 trials. If the first part of the action sequence, the *lock* action, failed in a trial, the second *slide-kick* action cannot be performed in that trial. Therefore, the trials shown as the horizontal in the figure do not include the trials with a failed *lock* action.

The difference in positions seems to be more difficult to reduce than the difference in angles in figure 11. Additionally, the position difference appears more noisy than the angle difference. The reason is that there are 3 motor primitives for the moving and 2 for the turning, as shown in figure 6. As we have argued, the performance of a motor primitive will be effected by the status of the motor on which the next motor primitive depends heavily. The learning of the moving is therefore slower and with more noises than the learning of the turning.

In order to give a concrete impression on how the human action is imitated, we show the moving and turning of Star-Kick together with the human action in Figure 12. The first row of the figures shows the trial at the very beginning of the learning excluding the *slide-kick* action. The second row of the figures shows one of the examples in which the ball was successfully kicked into the goal. In these figures, human actions are shown as a base line in solid. The point-lines depict the actions of Star-Kick. The learning algorithm moves the point-lines towards the solid lines over the trials.

As shown in the first row of the figures, the first guess of the motor speed is already very close to the target values, whereas the second and third commands do not work in a proper way. The ball is kicked from 500 to $600(ms)$ by the human player, and should be kicked within the same period by Star-Kick. The position difference at this period appears to be 20 to $40(mm)$ and the angle difference is 2 to $4(rad.)$, which means that the pose of the playing figure is nearly up-

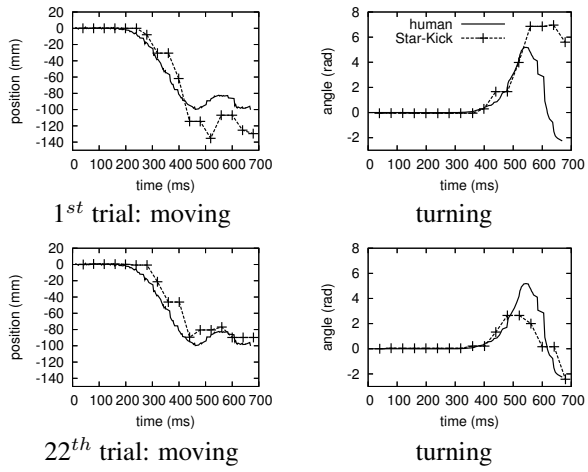


Figure 12: Different stages of the imitation

side down and the position is on average $30(mm)$ different from the ball. Obviously, the ball would not be touched at all in this situation.

The second row of the figure 12 shows that the 22^{nd} trial has a much smaller position and angle differences from 500 to $600(ms)$. However, at least one data point in "turning", which is near time point $550(ms)$, is still significantly different from the relative data point of the human action. The turning from 6 to $-2(rad.)$ during 550 to $700(ms)$ eventually kicks the ball with a fast speed in the *slide-kick* action of the human player. We observed in the experiments that the motor cannot reach a similar acceleration comparable to the movement of a human player even if the motor is set to turn in full speed. Due to this limitation of the motors, the human action cannot be imitated perfectly. An interesting point in our experiments is that although the action sequence of Star-Kick cannot be 100% similar to the human action sequence, the game ball can still be kicked into the goal.

Conclusion and Perspectives

In this paper, two independent learning algorithms are described that are able to refine an action sequence *lock* and *slid-kick* for the table soccer robot Star-Kick. One of them utilizes feedbacks of the performed actions. The other employs learning by imitation. Our approach allows for a scenario where the action sequence can be acquired while human players enjoy the games. We show in the experiments that the two algorithms can work together to improve the results of the action sequence. The learning process requires about 20 trials to reach a usable outcome, which happens when the *lock* action has a successful rate of approximately 75%, and the *slid-kick* action becomes a threatening attack, i.e. making a goal.

Our approach and experiments show a way to acquire a new action sequence in limited trials for the table soccer robot Star-Kick. We observed in the experiments that the ball can be effectively kicked into the goal even if the performed actions of the robot do not perfectly imitate the human actions. This observation indicates that human actions are suitable for Star-Kick to imitate with noise in a result-driven way, and learning by imitation is an effective approach to reduce the number of required trials.

We also trained two independent learning models with one learning process in our experiments. It indicates that separating and isolating learning problems into smaller models helps to improve efficiency in the learning process. The approach is practical for real robot systems such as Star-Kick.

In the future, we will explore miscellaneous human actions that Star-Kick might be able to imitate. A more complex action selection mechanism using policy gradient MDPs will be implemented to choose a proper action in different situations.

Acknowledgments

Star-Kick is a product of Gauselmann AG. Recording and imitating human actions in table soccer games was supported by a Karl-Steinbuch scholarship.

References

Bentivegna, D.; Ude, A.; Atkeson, C.; and Cheng, G. 2004. Learning to Act from Observation and Practice. *International Journal of Humanoid Robotics* 1.

Billard, A., and Siegwart, R. 2004. EDITORIAL: Robot Programming by Demonstration. *Robotics & Autonomous Systems, Special Issue on Robot Programming by Demonstration* 47/2-3.

Dillmann, R.; Rogalla, O.; Ehrenmann, M.; Zollner, R.; and Bordegoni, M. 1999. Learning Robot Behaviour and Skills based on Human Demonstration and Advice: the Machine Learning Paradigm. In *9th International Symposium of Robotics Research (ISRR 1999)*, 229–238.

foosbot. 2006. http://en.wikipedia.org/wiki/Foosbot.

Mataric, M. J. 2000. Getting Humanoids to Move and Imitate. *IEEE Intelligent Systems* 15(4):18–24.

Price, B. 2003. *Accelerating Reinforcement Learning with Imitation*. Ph.D. Dissertation, University of British Columbia.

Sammut, C.; Hurst, S.; Kedzier, D.; and Michie, D. 1992. Learning to Fly. In *The Ninth International Conference on Machine Learning*. In D. Sleeman and P. Edwards (Eds.). Aberdeen: Morgan Kaufmann.

Schaal, S. 1999. Is Imitation Learning the Route to Humanoid Robots? *Trends in Cognitive Sciences* 3.

Smart, W. D., and Kaelbling, L. P. 2002. Effective reinforcement learning for mobile robots. In *International Conference on Robotics and Automation*.

Weigel, T., and Nebel, B. 2002. Kiro – An Autonomous Table Soccer Player. In *Proceedings of RoboCup Symposium '02*, 119 – 127.

Weigel, T. 2005. Kiro – A Table Soccer Robot Ready for the Market. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 4277–4282.

Zhang, D. 2005. Action Selection and Action Control for Playing Table Soccer Using Markov Decision Processes. Master's thesis, University of Freiburg, Germany.