# Recording and Segmenting Table Soccer Games – Initial Results

**Dapeng Zhang[1] (zhangd@informatik.uni-freiburg.de), Bernhard Nebel[2] (nebel@informatik.uni-freiburg.de)**
[1,2] Research Group on the Foundations of Artificial Intelligence
University of Freiburg Germany D-79110

**Abstract**

Robot KiRo can play one side of a table soccer game autonomously. Our recent research focuses on learning from and acting against human actions. Therefore recording and segmenting games played by humans are motivated. In this paper, the construction of a table soccer game recorder is sketched. An intuitive segmenting algorithm is implemented to explore the properties of the recorded data. A segmentation approach using Hidden Markov Models (HMMs) is proposed.

## Introduction

Table soccer or "foosball" is a popular indoor sport. Similar to other ball games, it is a competition between two teams. Four game rods are mounted along the long sides of the table. During a table soccer game, players play the ball on a rectangular game field by using the figures mounted on the game rods. There is a goal at each end of the playing field. Each team tries to score goals while defending against attacks.

Playing table soccer requires fast reaction, accurate movements, and skillful actions. Starting from the initial situation with a stationary ball, professional players can perform the complex action sequence "slide-kick", which includes a "pass" and a "kick" to attack the goal, within 0.3 seconds, ending with the ball in the goal. The ball can easily reach a speed of $10m/s$ in the game.

Our robot KiRo can play one side of a table soccer game autonomously. It has won against most human players since 2002 (Weigel & Nebel, 2002). Cooperating with a company, we developed KiRo further, resulting in a commercial product in 2005 called *StarKick* (Weigel, 2005), which can win 90% of all the games against average human players. However, StarKick is completely reactive lacking any level of sophistication.

Our recent research focuses on learning from (Zhang & Nebel, 2007) and acting against human actions, as well as attacking the weakness of human players. For this purpose, we are designing mechanisms for recording and segmenting games played by humans, which we describe here.

In the next section, the construction of a table soccer game recorder is sketched. We implemented a practical and easy approach to obtain accurate measurements with high frequency from a game table. Based on that, we describe a finite state machine, which we use to segment the recorded data. It is an easy to implement and intuitive approach that explores the properties of the recorded data. We show in a small experiment that this approach works already reasonably, but it has some limitations. In the last section, an approach using Hidden Markov Models (HMMs) is proposed, which can overcome the limitations of the existing method and which can provide a basis for the prediction of the human actions.

## Recording Games

Recording table-soccer games requires estimating the state of the game rods, i.e., angle and position, and estimating the position of the ball. An overhead camera is used in KiRo to observe the ball with a frame rate of $50Hz$ (Weigel, Zhang, Rechert, & Nebel, 2004). Eight DC motors are used to control four game rods of one side in KiRo. The positions and the angles of the computer controlled rods are provided by the motor encoders. The angle and position of the rods controlled by the human are only very crudely estimated using the camera. The existing system can be used for recording, however, the accuracy of the sensors, the speed of the movements, and the required processing power are challenges. Besides, ball detection is hard if it is hidden by a playing figure. For example, if we want to use the overhead camera to record the "slide-kick" action sequence which is mentioned in the beginning, there are only 6 frames to record this action sequence. It would be very hard to either imitate or explain the action sequence based on these 6 frames (Zhang & Nebel, 2007).

In this approach, we use a laser range finder to estimate the position of the ball, a rotary encoder and a distance sensor to estimate the movement of each rod. As it is very expensive to build sensors and mechanical parts from scratch, our recorder is mainly made from parts available on the market. Thus, our game recorder balances costs and quality.

As the basis of our development, a game table was purchased. We have chosen the *SICK LMS400* laser scanner for observing the ball. It has a frame rate of $350Hz$ and an angular resolution of 0.25 degrees, which means that the data is much better than anything an ordinary camera can provide. As there is a gap of about $12mm$ between the playing surface and the end of a playing figure, and as the valid range of the sensor is from $700mm$ to $3000mm$, we set *LMS400* behind the goal of the game table so that the laser beam can go through the gap, and shoot on the lower part of the ball without any disturbance from the playing figures.

The linear position of a game rod is estimated by a *SICK DT20* sensor. It is an optical distance sensor which has an accuracy of $1.5mm$ and it can measure the distance without any physical contact to the target. A rotary encoder *BPSD25* is used to observe the turning of the rod. It is a single-turn encoder with an angular step of 0.043 degrees. A slide-joint unit is installed between the game rod and the rotary encoder. Thus the game rod can be moved as usual along the length

direction while the turning is transferred to the rotary encoder.

We implemented a calibration program, which creates a two dimensional coordinate system linked to the playing surface. Consequently, the raw sensor data can be transferred to the coordinates. As explained, the laser beam of *LMS400* is reflected by the lower part of the ball. The walls around the playing field form the "background" for the laser beam. The information about the ball position can be estimated by removing the background from the raw sensor data. And the coordinates can be calculated by measuring the distance between the ball and the boundaries. The calibration process for the *LMS400* has to detect the the boundaries or the background, which are the border lines matching the data points of *LMS400* as well as possible. An algorithm was implemented to project all data points onto a line. If the line is perpendicular to the target boundaries, there should be a peak of projected data points for each boundary. The calibration algorithm thus has to search for a line which leads to the highest peaks of the projected data points.

Calibrating the distance sensor is simple, the maximum and minimum data points are recorded, which requires moving the game rod to each end manually. The position of the rod can be calculated by a linear function. Similar to the calibration of the distance sensor, the game rod is set upright manually in the calibration process of the rotary encoder. There the sensor data is recorded as zero. The rod angle can be calculated as the offset from the recorded zero position later.

We installed the distance sensors and rotary encoders on the attacker rod of one team, the red team, and on the defender and goalkeeper rods of the other team, the blue team. The accuracy of the data is checked by putting the ball at several specific points on the playing surface, such as the center point or the corners of the field lines, and observing the relative position of the ball and a game rod. The sensor data is visualized on a screen, matching the physical situation. A sequence of actions which includes several dribble and "slide-kick" actions was recorded and saved as a log file. The sequence lasted for 46.62 seconds, during which there were 13004 ball position estimations, 13644 rod positions and 9816 rod angle estimations. Thus the frame rate is more than 210Hz.

## Segmenting the Games – Initial Results

When a human is playing table soccer, the intentions are typically "stopping", "dribbling", or "attacking". "Stopping" means getting control of the ball. "Dribbling" is a preparation for "attacking" during which the ball is passed to a comfortable position, which is always the start point of an "attacking". "Attacking" contains a sequence of action, such as a "slide-kick", ending at kicking the ball towards the goal. In this paper, the tasks of segmentation is defined as segmenting the recorded sensor data according to the three intentions defined above.

There are several challenges in the segmentation tasks. First, we need a bridge between the inputs and the outputs. The input of the segmenting system is a sequence of coordinates and angle values with time-stamps, while the output labels, "stopping", "dribbling", and "attacking" are abstract symbolic data. Second, the segmentation should balance efficiency and quality. The sensors provide high-frequency data with noises, nevertheless the segmentation should work in an on-line manner, being tolerant to noise. Creating a data set, which is sufficient for training the target model, is another possible challenge. Training a segmentation model normally requires a training set, helping a machine learning algorithm to adjust the parameters of a model automatically.

In this section, we describe the implementation of an intuitive and easy to implement segmentation algorithm using finite state machines. A small experiment is done in order to investigate the properties of the recorded data, and set a base line for further experiments.

Many machine learning algorithms are based on features with discrete values. Our basic idea is to transform the recorded coordinates and angles to a set of discrete features. An "active" figure is defined as the figure which is controlling the ball or going to control the ball. The spatial relations between the game ball and the active figure is the most important feature in our implementation, in which 13 relations are defined. They are *left-align*, *right-align*, *forward-align*, *backward-align*, *left-touch*, *right-touch*, *forward-touch*, *backward-touch*, *positive-lock*, *negative-lock*, *left-detach*, *right-detach*, and *irrelevant*. Figure 1 shows the first eight of these relations. *Left-detach* and *right-detach* means that the ball and the active figure are not aligned in any direction. *Irrelevant* means the ball is not within the control range of the active figure.

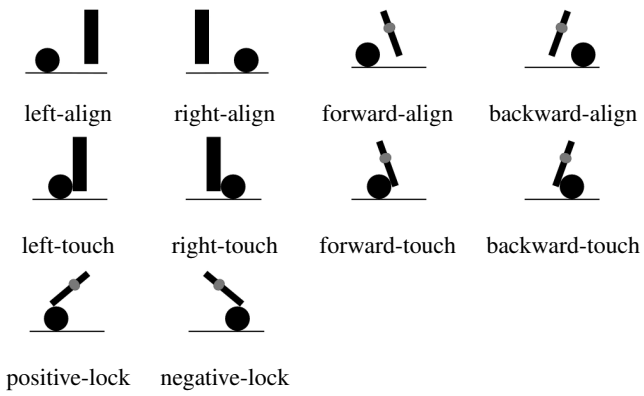


Figure 1: Spatial Relations

So far, an action sequence can be transformed into a sequence of states. We merge the successive data-slices together if they have the same spatial-relation value. For example a "slide-kick" action could start at *positive-lock*, pass *left-detach*, *left-align*, *left-detach*, *forward-align*, *forward-touch*, and finally end at *forward-align*. Each relation mentioned here could include several data-slices, and extra features such as start and end time-points are added to create a "state". Based on the states, a set of finite state automatons were constructed, each of them can recognize one of the following primitive actions: "active-rod changed", "stop", "touch", and "kick". "Active-rod changed" is the situation where the ball moves from one game rod to another. "Stop" means the ball is either locked or still for a short time span. "Touch" and "kick" are the actions where the ball is either side-touched or kicked by the active figure. We added some limitations on the duration of a primitive action which reduces the noise and gets rid of actions that are too slow. Based on the ability to recognize primitive actions, the segmentation was finally

implemented according to the following rules. "Stopping" starts at an "active-rod changed" and ends at a "stop". "Dribbling" contains two or more succeeded "stop". "Attacking" should start at a "stop", pass a "kick", and end at a "active-rod changed".

The experiments are based on the mentioned log file. We run the algorithm to segment the log data into target action sequences. The segmented data were replayed and checked manually. We did not find any intended sequence missed or any unintended sequence appearing in the results. The used computer has a $3.2GHz$ CPU and 1G memory. The log file was played in its real speed. The algorithm finished the tasks as soon as the log playing stopped. The CUP monitor did not show any processing pressure during the process.

## Discussion

Our segmenting algorithm uses a layered structure to build a bridge from the "low-level" sensor data to "high-level" action sequence labels. The noise is filtered out by using domain knowledge, which is mainly expressed as the spatial relations and the duration of the actions. Although the segmentation works fine with the log file, it also shows some limitations. For example, the algorithm which maps continuous coordinates and angles to spatial relations has value oscillating problems near boundaries. Many parameters are defined manually, which makes it hard to maintain it in the future. It is not easy to extend the approach for other interesting topics such as an on-line prediction. To overcome the shortcomings of the existing approach, Hidden Markov Models (HMMs) appear to be one of the interesting solutions.

HMMs are based on Markov chains and statistics (Baum & Petrie, 1966). They were first successfully used in speech recognition (Baker, 1975) and have become wide spread in many domains, for example, HMMs can be used to model DNA sequences in bio-informatics (Kulp, Haussler, Reese, & Eeckman, 1996). They are employed to describe the return series in finance (Rydn, Tersvirta, & sbrink, 1998) and they can also help to navigate a robot (Shatkay & Kaelbling, 1997).

There are five elements in an HMM: a set of states, a set of observations, probability distribution to describe the state transition, probability distribution of the observations per state, and probability distribution of the initial state. Observation is an available element in an HMM, which could be either with discrete or with continuous values. Methods were developed to learn the other four elements automatically. The Baum-Welch algorithm computes the maximum likelihood of probability distributions in an HMM using forward-back algorithm (Bilmes, 1997). Kohonen network clustering can be employed to learn the states of an HMM (Fox, Ghallab, Infantes, & Long, 2006).

As the existing learning algorithms for HMMs search for a local maximum of the likelihood, it would be interesting to try and compare the following two approaches using HMMs in the context of our research. On the one hand, the approach using the fully automated method (Fox et al., 2006) could be implemented. On the other hand, HMMs are constructed by using known information as much as possible. In this approach, the states of HMMs can be defined using the described spatial relations. Based on the knowledge of the existing transition models (Rabiner, 1989), we could choose a left-right model in our implementation. The observations are with continuous values. The probability distributions of the HMMs can be initialized by some supervised learning approaches. The states could be refined using the Viterbi algorithm. And finally the Baum-Welch algorithm can be used to enforce convergence of the parameters.

We will report on the success of these approaches in a follow-up paper.

## References


Baker, J. K. (1975). The Dragon System – An Overview. *IEEE Trans. Acoust. Speech Signal Processing*, *ASSP-23*, 24-29.

Baum, L., & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Ann. Math. Stat.*, *37*, 1554-1563.

Bilmes, J. (1997). *A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models* (Vol. ICSI-TR-97-021; Tech. Rep.). University of Berkeley.

Fox, M., Ghallab, M., Infantes, G., & Long, D. (2006). Robot introspection using learned hidden markov models. *Artificial Intelligence*, *170*(2), 59-113.

Kulp, D., Haussler, D., Reese, M. G., & Eeckman, F. H. (1996, June). A Generalized Hidden Markov Model for the Recognition of Human Genes in DNA. In *Proceedings of the fourth international conference on intelligent systems for molecular biology* (p. 134-142). Menlo Park, CA: AAAI Press.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rydn, T., Tersvirta, T., & sbrink, S. (1998). Stylized facts of daily return series and the hidden markov model. *Journal of Applied Econometrics*, *13*(3), 217-244.

Shatkay, H., & Kaelbling, L. P. (1997). Learning topological maps with weak local odometric information. In *Proceedings of the fifteenth international joint conference on artificial intelligence (ijcai)* (p. 920-929).

Weigel, T. (2005). Kiro – A Table Soccer Robot Ready for the Market. In *Proceedings of the ieee international conference on robotics and automation (icra)* (p. 4277-4282). Barcelona, Spain.

Weigel, T., & Nebel, B. (2002). Kiro – An Autonomous Table Soccer Player. In *Proceedings of robocup symposium '02* (pp. 119 – 127). Fukuoka, Japan.

Weigel, T., Zhang, D., Rechert, K., & Nebel, B. (2004). Adaptive Vision for Playing Table Soccer. In *Proceedings of the 27th german conference on artificial intelligence* (pp. 424–438). Ulm, Germany.

Zhang, D., & Nebel, B. (2007, June). Learning a Table Soccer Robot a New Action Sequence by Observing and Imitating. In *Proceedings of the 3rd artificial intelligence for interactive digital entertainment conference (aiide)* (p. 61-67). Stanford, California.