Behavior Recognition and Opponent Modelling for Adaptive Table Soccer Playing

Thilo Weigel, Klaus Rechert and Bernhard Nebel

Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
weigel,rechert,nebel@informatik.uni-freiburg.de

Abstract. We present an approach for automatically adapting the behavior of an autonomous table soccer robot to a human opponent player. For this, basic actions are recognized as they are performed by the human player and characteristic action observations are used to establish a model of the opponent. Based on the model, the opponent's playing skills are classified with respect to different levels of expertise and its particular offensive and defensive skills are assessed. In response to the knowledge about the opponent, the robot adapts the velocities at which it attacks and defends in order to provide entertaining games for a wide range of human players with different playing skills. Experiments on two different table soccer robots validate our approach.

1 Introduction

For an autonomous system, the best strategy for achieving a certain goal often depends on the behavior of other agents. As the agents usually differ in their behavior patterns, the ability to adapt to these differences dynamically is very beneficial for optimizing the agent's behavior.

Reinforcement learning is a common technique for adapting an agent's policy to the environment [1]. As this is done in a trial-and-error fashion, an agent generally doesn't derive explicit knowledge about the encountered agents, but rather implicitly learns to act in the most appropriate way. Unfortunately, in realistic environments reinforcement learning approaches are usually too slow to be used for online adaption.

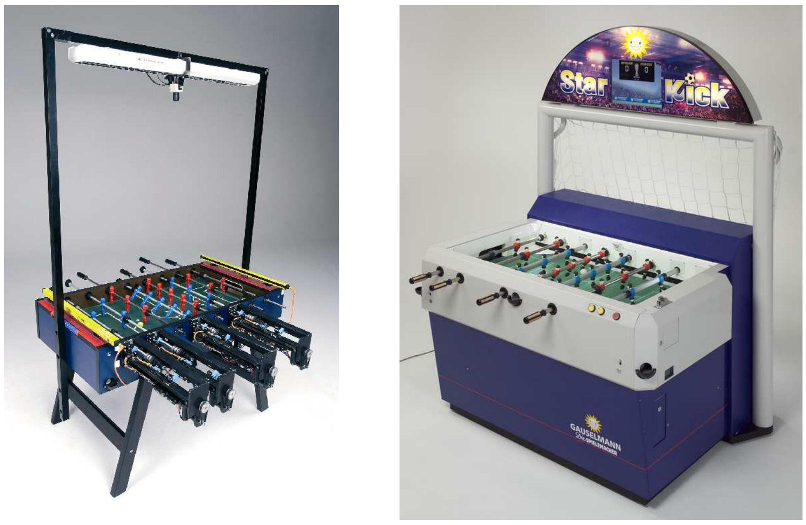
In contrast, deliberate modification of an agent's behavior based on recognized features of other encountered agents allows to adapt in a much more efficient way. However, this requires to explicitly gather and classify information about other agents' behavior patterns. In game playing, one would like to obtain a model of the opponent which characterizes its playing style, playing skills and general strategy. Based on that model, the agent then selects a specific strategy which promises to be the optimal response to the opponent.

In this paper, we present an approach for the automated adaption of a table soccer robot to its human opponent[1]. The basis for the approach is the robust vision-based recognition of basic skills as they are performed by the human players. According to

[1] Table soccer is also commonly known as *foosball*.

the observations, the opponent is classified with respect to different opponent models. In consequence, the robot adapts its behavior taking into account the implications of the most likely model.

The approach is evaluated on two variants of a real table soccer robot. A first prototype version called *KiRo* uses an overhead color camera for observing the players and the ball [2]. A commercial version called *StarKick* uses a black and white camera perceiving the ball from underneath the table. In many test games StarKick showed to be a competitive challenge even for advanced human players [3]. Figure 1 shows pictures of KiRo and StarKick.



(a) (b)

Fig. 1. The table soccer robots KiRo (a) and StarKick (b).

The benefit of adapting to the opponent is twofold. On the one hand, weaknesses of the opponent can be exploited, but on the other hand, the playing level can be adjusted so that the game stays interesting for less experienced players, as well.

In contrast to the work presented in this paper, many related approaches have only been evaluated in simulated environments. In the context of simulated robotic soccer, basic soccer skills were formalized and recognized using Hidden Markov Models [4]. For representing different classes of adversaries, it was proposed to capture the opponent strategic behavior by accumulating position information in grids and to use decision tree learning for classifying the opponent according to similarities of these grids [5]. The *ATAC* approach models the opponent team behavior by probabilistic representations of the opponents' predicted locations. The models have been successfully used to adapt coordinated team plans for setplays [6].

The rest of this paper is structured as follows. In Section 2 the recognition of basic actions based on vision data is described. Section 3 presents how the observed actions and action parameters are used to derive a model of the opponent's playing style and playing skills. In Section 5 we discuss experimental results and we conclude with Section 6.

2 Behavior Recognition

The basis for behavior recogniton are camera images as shown in Figure 2. The images are delivered by the vision system in YUV-format at 50 Hz[2] with a resolution of 384×288 pixels. For StarKick, the possibility to detect the playing figures in the camera

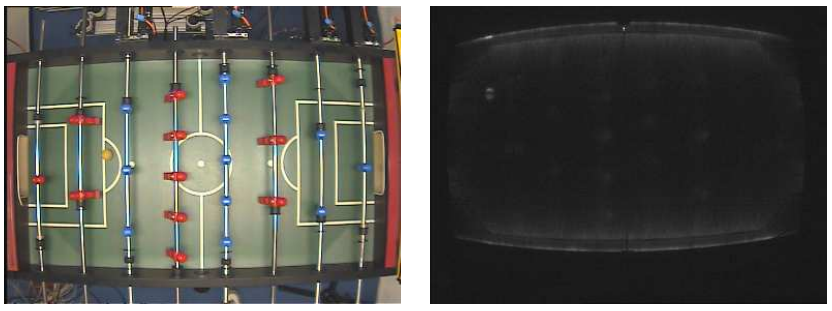

(a) (b)

Fig. 2. Camera images delivered from KiRo's overhead camera (a) and StarKick's camera underneath the table (b).

images was traded for an infrared-based system which allows a very robust and reliable ball detection. Only the own player position are available using motor controller feedback [3]. Even though all player positions can be extracted from KiRo's camera images, it is in general not possible to reliably detect which playing figure caused a certain ball movement: Even at normal game velocities the ball can travel several centimeters between two consecutive camera frames and due to the limited image resolution the ball can be recognized only with an accuracy of 3mm at best [7].

For these reasons, we decided to base our approach solely on the observed ball position during a game and the knowledge of the rods' fixed axis locations. The ball's heading and velocity is determined from consecutive ball estimates which are maintained using a standard Kalman Filter.

In order to cope with the limited spatial and temporal resolution of the available data, we chose a very coarse representation for defining and recognizing the most common actions during a table soccer game. For each player p and rod r, we introduce a corresponding *influence area* $\mathcal{I}_{p_r}$ in which at least one of the rod's playing figure can manipulate the ball. Figure 3 shows the corresponding coordinate system and the influence areas for the eight rods.

An action is now defined for an entire rod based on where the ball enters its influence area, how it moves inside and where it leaves the influence area again. We distinguish between the following possibilities for manipulating the ball:

[2] This is achieved by processing each half-frame individually.

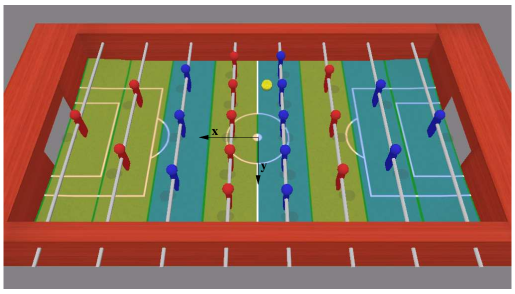


Fig. 3. The influence areas marked with different colors for the red and blue rods.

- **BlockBall**
  A rod is said to have blocked the ball, when it prevented the ball from passing without gaining control over it. This is expressed by the fact that the ball stayed inside the influence area only for a short amount of time and it left the area at the same side it entered.
- **ControlBall**
  A rod is said to have controlled the ball, when it kept the ball motionless inside the influence area for a significant amount of time.
- **DribbleBall**
  A rod is said to have dribbled the ball, when it kept the ball inside the influence area for a significant amount of time and the ball moved over a significant distance.
- **KickBall**
  A rod is said to have kicked the ball, when it accelerated the ball considerably. A kick can either reverse the ball's trajectory or continue it when the ball arrived from behind. After a dribble or control action, it is always assumed that a kick causes the ball to finally leave the influence area.
- **YieldBall**
  A rod is said to have yielded the ball, when the ball traversed the influence without a significant change in velocity and orientation. This action may reflect the intention to let the ball pass from behind, but also may be observed when an intended block failed.

Here, the actions for blocking, kicking and yielding the ball refer to both directions the ball can traverse an influence area. However, the direction of the ball is explicitly considered when the opponent's playing skills are assessed.

The above actions can be described more formally by a set of predicates which are all based on directly observable features. Let $(x_t, y_t, v_t, \alpha_t)$ be the ball vector at time t with (x_t, y_t) denoting the ball's position, v_t its velocity and α_t its heading. Let further be $t = 0$ the time the ball entered the influence area $\mathcal{I}_{p_r}$[3]. We can then formally define

[3] According to the camera's frame rate each time step currently corresponds to 20 milliseconds.

the following predicates for the r-th rod of player p:

TimeOut: $$t > \frac{\Delta_{rod}}{\cos \alpha_0 v_0} \quad (1)$$

BallMoved: $$\frac{1}{t} \sum_{i=0}^{t} \sqrt{x_i^2 + y_i^2} > \Delta_d \quad (2)$$

BallLeft: $$x_t \notin \mathcal{I}_{p_r} \quad (3)$$

BallAccelerated: $$v_t > a v_0 \quad (4)$$

BallDirectionChanged: $$\operatorname{sgn}(X_{p_r} - x_0) = \operatorname{sgn}(X_{p_r} - x_t) \quad (5)$$

Here, Δ_{rod} denotes the distance between two neighboring rods. In order to capture, when a ball was stopped by a rod, *TimeOut* checks, whether the current time t exceeds the time, after which the ball is expected to leave the influence area again. In practice, a timeout is only calculated if $|\alpha_0| > 90° \wedge v_0 > 0$ is true. Furthermore, a lower and upper bound limit the timeout to reasonable values. In our experiments, timeouts in the interval $[200 msec, 750 msec]$ gave very satisfying results. The ball's movements inside an influence area are summed up by the predicate *BallMoved* which is true when per time step the ball traveled more than a certain minimum required distance. To detect if the ball hast left the influence area, *BallLeft* checks, whether the ball coordinates still belong to that area. The predicate *BallAccelerated* determines, if the ball's initial velocity increased considerably. In practice, the ball is only considered to be deliberately accelerated by a player, if the following constraints hold for the initial and final ball velocity: $v_0 < v_{max} \wedge v_t > v_{min}$. In our experiments, we achieved good results with $v_{min} = 350 \frac{mm}{s}$ and $v_{max} = 5000 \frac{mm}{s}$. The predicate *BallDirectionChanged* checks, if the ball left the influence area to the same direction from where it entered. The distance threshold Δ_d and the acceleration factor a can be determined empirically. However, in the future we plan to learn the their values automatically from training data.

The basic actions can now be defined using the above predicates: An action is considered to be recognized if the conjunction of some of the – potentially negated – predicates is true. The decision tree shown in Figure 4 represents these formulas and detects an action for the rod whose influence area currently contains the ball.

The decision tree is evaluated in every cycle. An action can be classified either when the ball changed to another influence area or when a timeout occurred. If both *BallLeft* and *Timeout* are false, no action is classified. In case no classification is made, one could in principle try to estimate the action currently taking place. However, as an action usually happens within only a few cycles, predictions ahead of time are generally very uncertain.

The predicate *BallAccelerated* reflects, if a kick action occurred. For dribble and control actions the corresponding initial ball velocity is set to zero: $v_0 = 0$. This way, after a control or dribble action, always a kick is classified as well. Please note, that the time t is reset to zero after an action is recognized and thus consecutive dribble or control actions may be detected. Such sequences are merged to one single action.

Figure 5 shows an example series of camera images while the blue attacker is kicking the ball. As can bee seen, the kicking action takes place within only a few milliseconds.

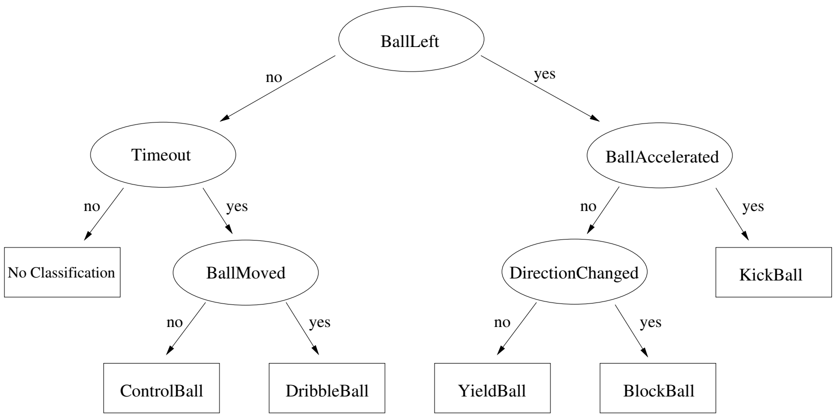


Fig. 4. The decision tree for classifying the basic table soccer actions.

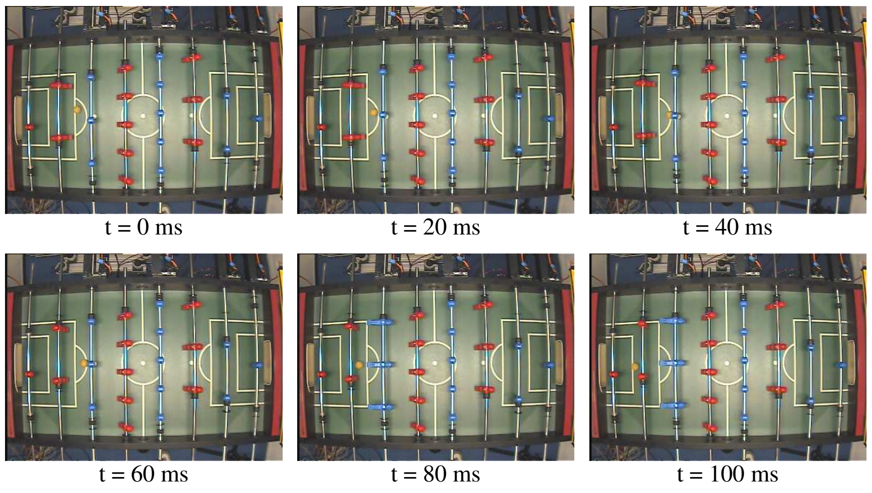


Fig. 5. Camera images in a 100 ms period during a kick action of the blue attacker.

In some special cases, the described method does not yield the desired action classifications. However, using specific domain knowledge, some observations can be filtered and some can be created artificially for improving the classification results.

Since the goalkeeper's movements are limited to the area in front of its goal, it cannot reach the lateral positions of its influence area. In consequence, when a ball is bounced back from the goal border, a block action would be assessed even though it is impossible that the goalkeeper actively blocked the ball. In such cases, the corresponding observations are filtered and no action is classified.

When the ball moves very fast, it may cross entire influence areas inbetween two consecutive camera frames so that no ball observations are available for these areas. As in such situations the corresponding rods obviously let the ball pass, appropriate observations are generated artificially so that *YieldBall* actions are assessed for these rods.

Another special situation occurs, when a goal was shot and the ball is thrown in again from the center of the horizontal table border. In these situations, the ball does not traverse neighboring influence areas and thus no yield actions should be classified. Such situations can be detected reliably by checking if the ball was last observed in a defender's or goalkeeper's influence area and then (re-)appeared in a rectangle around the regions where the ball can be thrown in. In consequence, the above heuristic is suspended, the ball's initial velocity v_0 is assumed to be zero and the allowed timeout is initialized with a fixed value. This way, after a throw in, always a *KickBall* is assessed, but a *ControlBall* or a *DribbleBall* are only classified when there was a certain time delay before the kick action.

3 Opponent Modelling

A player's expertise and general playing style can be assessed based on the observed frequency and characteristics of the recognized basic actions. For this, a set of parameters is observed and evaluated for each occurrence of a basic action. We distinguish between the same actions as presented in section 2. Each observation is rated with respect to a predefined standard yielding a value in the interval $[0, 1]$. For example, the rating of the ball's velocity v_t after a kick action is mapped to $[0, 1]$ by the following function:

$$r_{kick}^{vel} = \begin{cases} 0, & \text{if} \quad v_t < v_{min} \\ \dfrac{v_t - v_{min}}{v_{max} - v_{min}}, & \text{if} \quad v_{min} < v_t < v_{max} \\ 1, & \text{else} \end{cases} \tag{6}$$

Here, v_{min} and v_{max} denote predefined minimum and maximum velocities, which were set in our case to $300\frac{mm}{sec}$ and $2000\frac{mm}{sec}$, respectively.

With r_a^o denoting the rating of an observation o for the action a and $action$ denoting a placeholder for any possible action, the following observation ratings are taken into account:

- r_{action}^{num} – The percentage of occurences of an action with respect to the total number of observed actions so far
- r_{action}^{vel} – The ball's maximum velocity while an action took place
- $r_{dribble}^{distance}$ – The distance the ball moved during *DribbleBall*
- r_{kick}^{gain} – How far the ball was shot by *KickBall* relative to the distance it could be shot until reaching the opponent goal

The observations can directly be related to a player's *level of expertise*. Other observations like the time the ball was kept inside an influence area by a control action, or the distance the ball traveled during a dribble action would not necessarily reveal, how well the opponent plays, but rather show the opponent's *playing style*.

Two additional observations can be interpreted as observing two higher level actions:

– **MoveKick**
  A *MoveKick* is performed, when a rod moves the ball sideways with considerable speed and then kicks the ball forward immediately. This action is observed, when a fast *DribbleBall* is directly followed by a *KickBall*.
– **ControlledDribble**
  A *ControlledDribble* is performed, when a rod controls the ball right after it dribbled the ball. This action is observed, when at least one time a *DribbleBall* is immediately followed by a *ControlBall*.

As these actions usually require advanced playing skills, they are helpful for distinguishing between unexperienced and more advanced players.

The ratings of the different observations for one action are now combined to one quality measure for that particular action. As the rating of an action's occurrence frequency does not relate to an individual action but rather provides an additional global quality measure, it is incorporated at a later stage. In fact, for *Block*, *Control*, *Yield* and *ControlledDribble* actions, only their occurrence frequency is of interest[4]. Therefore, only the following quality measures have to be calculated as the weighted sum of observation ratings:

$$q_{dribble} = \frac{w_1 r^{vel}_{dribble} + w_2 r^{distance}_{dribble}}{w_1 + w_2} \tag{7}$$

$$q_{kick} = \frac{w_3 r^{gain}_{kick} + w_4 r^{vel}_{kick}}{w_3 + w_4} \tag{8}$$

$$q_{moveKick} = \frac{w_5 r^{gain}_{moveKick} + w_6 r^{vel}_{moveKick} + w_7 r^{vel}_{dribble}}{w_5 + w_6 + w_7} \tag{9}$$

where the $w_{\{1...7\}}$ denote some weights.

Usually, an action is carried out by a player with different quality levels during a game. The performance of a player may sometimes be very bad due to mistakes and sometimes be very good just by luck. In order to capture a player's general playing skills, we therefore eliminate such "outliers" and assume, that a player's real capabilities are best reflected by the quality measure which occurred the most times. For this, we discretize the quality measures corresponding to one action into *quality levels* and build a histogram over these levels. The maximum of a histogram is then taken as the overall quality measure. Figure 6 shows such a histogram for the kick qualities of three different players and illustrates, how better players achieve higher qualities.

The playing skills of an opponent player can now be assessed based on the actions' quality measures and occurrence frequencies. We distinguish between general skills for blocking, kicking and controlling the ball and additionally consider if a player is

[4] For block actions, the velocity at which the ball approaches a rod could be taken as a quality measure. However, the quality would then depend very much on the opponent's kicking skills.

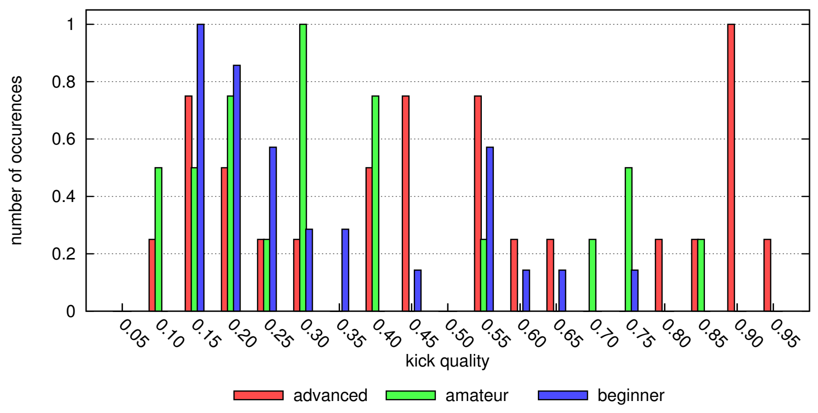


Fig. 6. The normalized number of occurrences of the kick qualities for three human players with different levels of expertise.

capable of performing the more difficult actions *ControlledDribble* and *MoveKick*:

$$\mathcal{S}_{Kick} = \hat{q}_{kick} \tag{10}$$

$$\mathcal{S}_{Block} = \frac{w_1 r^{num}_{block} + w_2(1 - r^{num}_{yield})}{w_1 + w_2} \tag{11}$$

$$\mathcal{S}_{Control} = \frac{w_3\hat{q}_{dribble} + w_4 r^{num}_{dribble} + w_5 r^{num}_{control} + w_6(1 - r^{num}_{lost})}{w_3 + w_4 + w_5 + w_6} \tag{12}$$

$$\mathcal{S}_{BontrolledDribble} = r^{num}_{controlledDribble} \tag{13}$$

$$\mathcal{S}_{MoveKick} = \frac{w_7\hat{q}_{moveKick} + w_8 r^{num}_{moveKick}}{w_7 + w_8} \tag{14}$$

Here, $\mathcal{S}_s$ denotes a certain skill s, $\hat{q}_a$ denotes the maximum quality extracted from the histogram for action a and $w_{\{1...8\}}$ are weights (different to those used in formulas 7 – 9). Since even beginners kick the ball very often while more experienced players may choose to control the ball more often, we decided to ignore the occurrence frequency when assessing the kicking skills. In order to capture a players defensive skills, for block actions only the ones with the ball rolling towards the own goal are counted. Assuming that a player always at least intends to block the ball, additionally the occurrence percentage of yield actions which let the ball pass contrary to the own playing affect the blocking skills. The control skill summarizes a player's ability to deliberately control the ball and consequently, the occurrence frequency of dribble and control actions as well as the dribble quality is relevant. Additionally, the number of times the ball was lost is considered. This is the case whenever the velocity of a kick after a dribble or control action is rated as zero. Only low ratings r^{num}_{lost} can yield high control skills, indicating that most of the times the ball was deliberately played rather than accidentally lost.

An opponent model for assessing a player's level of expertise can now be defined as a vector containing the five skill measures:

$$\mathcal{O} = (\mathcal{S}_{Kick}, \mathcal{S}_{Block}, \mathcal{S}_{Control}, \mathcal{S}_{ControlledDribble}, \mathcal{S}_{MoveKick})^T \tag{15}$$

The state chart in Figure 7 shows the allowed action sequences while the action parameters of one team are extracted. The state chart validates a sequence and rejects it, if it is not possible that the basic actions are followed by each other in that way. The states *Receiving*, *Dribbling* and *Kicking* are reached, when an action moved the ball to a rod of the own team. Whenever a state transition takes place, the parameters of the action which caused the transition are updated. The state *OpponentHasBall* illustrates, how control of the ball shifts to the opponent team. The opponent team, in turn, maintains another state chart for the own parameter extraction. The state transitions are labeled with actions, that may cause the transition. Bold letters indicate actions of the own team and actions printed in grey show actions of the opponent. If an action is carried out by the same rod, it is plotted in italics. An action is displayed in normal letters, if it shifts the ball control to another rod. For further processing, consecutive controlled dribble and dribble actions are merged to one single controlled dribble action.

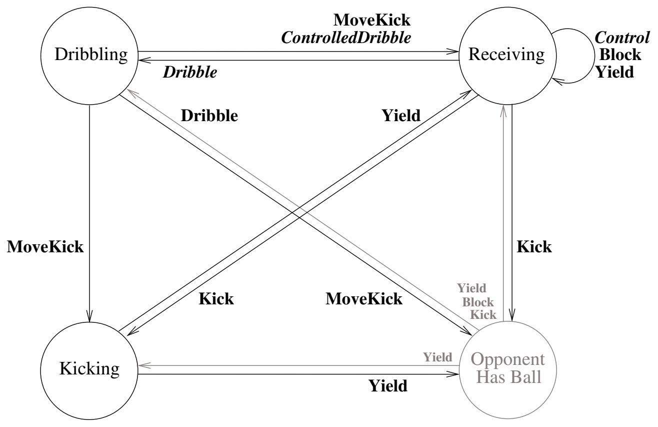


Fig. 7. The state chart for modeling the opponent.

A typical game sequence could look like the following: The opponent attacker shoots the ball past the own defender towards the own goalkeeper. The goalkeeper blocks the ball such that it rolls back to the own defender which now stops the ball. The defender then starts to move the ball inside its influence area, stopping it once in a while. Eventually, it shoots the ball forward past the opponent attacker and the own midfield, until the opponent midfield finally stops the ball. In the state chart, this sequence would start in the *OpponentHasBall* state, from where the opponent kick caused a transition to the *Receiving* state. The block action would lead again to the *Receiving* state. The alternating dribble and control actions would shift between the *Dribbling* and *Receiving* states, which would be merged to one single controlled dribble action. If the ball moved before it was kicked by the defender, a move kick action would transfer from the *Dribbling* to the *OpponentHasBall* state. Otherwise, a regular kick would transfer from *Receiving* to *OpponentHasBall*. As the opponent attacker lets the ball pass trough, the kick action continues and the *Kicking* state is reached again. However, the own midfield rod yields the ball, too, and the *OpponentHasBall* state is finally reached.

A resulting model vector describing an observed opponent can now be classified with respect to predefined opponent classes. In our case, these classes describe different levels of playing expertise and are defined by model vectors with fixed quality measures. In our experiments, we used the following definitions:

– $\mathcal{O}_{beginner} = (0.4, 0.3, 0.2, 0.0, 0.0)^T$
– $\mathcal{O}_{amateur} = (0.6, 0.5, 0.3, 0.2, 0.3)^T$
– $\mathcal{O}_{advanced} = (0.8, 0.8, 0.8, 0.8, 0.7)^T$
– $\mathcal{O}_{expert} = (1.0, 1.0, 1.0, 1.0, 1.0)^T$

For comparing an observed opponent model M with an opponent class C, the match between M and C is defined as the sum of the squared differences between the corresponding skill measures:

$$match(M, C) = \sum_{a \in \mathcal{A}} (\mathcal{S}_a^C - \mathcal{S}_a^M)^2, \tag{16}$$

where $\mathcal{A} = \{kick, block, control, dribble, controlledDribble, moveKick\}$.

Classifying a model vector M can now be considered as finding the class C with the minimum deviation to M. Thus, minimizing $match$ yields the desired classification:

$$class(M) = \arg\min_{C \in \mathcal{O}} match(M, C), \tag{17}$$

where $\mathcal{O} = \{\mathcal{O}_{beginner}, \mathcal{O}_{amateur}, \mathcal{O}_{amateur}, \mathcal{O}_{amateur}\}$. The function $class(M)$ assesses the general level of expertise of an observed opponent player. This information can be used in the following for adapting the own behavior appropriately.

4 Behavior Adaption

An agent can utilize the knowledge of the opponent's playing skills in various ways. One goal could be to exploit weaknesses of the opponent for increasing the own playing performance. Against weaker opponents, a goal could also consist in lowering the own playing standard to the opponent level for keeping the game interesting.

Since StarKick is capable of beating even advanced human players, its playing level usually needs to be lowered in order to maintain the game entertaining for the average human player. This can be achieved by lowering the velocities at which the robot shoots the ball and at which it moves a rod towards a certain blocking position. While the first weakens the robot's offense, the latter decreases its defensive play.

A more elaborate way of adaption would consist in employing different types of actions according to the opponent's playing characteristics. Currently, we are working on actions for dribbling and passing the ball. These actions provide a very attractive game to watch but also have a higher risk of loosing the ball to the opponent. Therefore, the use of these actions could be made dependend on the risk that an opponent would take too much advantage of them.

At present, we adapt the own behavior based on adapting the move and shoot velocities such that the game maintains balanced between the robot and the human opponent. Based on the opponent model proposed in the previous section, there are two alternatives for doing so. Both approaches adjust the velocities by multiplying constant factors $f_{move}, f_{turn} \in [0, 1]$ to the maximum velocity at which a rod moves and turns.

One way to establish these factors is to refer to predefined values according to the assessed opponent playing level. Table 1 shows the mapping from opponent playing level to velocity factors which we used in our experiments.

	Beginner	Amateur	Advanced	Expert
f_{move}	0.4	0.6	1.0	1.0
f_{turn}	0.2	0.5	0.8	1.0

Table 1. The factors for the maximum move and turn velocity with respect to the opponent's level of expertise.

Instead of only considering the general level of expertise, the opponent's defensive and offensive capabilities can be assessed individually for adapting in a more direct way to its strengths and weaknesses. A second adaption scheme therefore adjusts the maximum move velocity in response to the opponent's offensive skills and the maximum turn velocity according to the opponent's defensive skill level. For this, from the model vector an opponent's defensive and offensive skill level is calculated as follows:

$$\mathcal{S}_{defense} = \frac{1}{3}(\mathcal{S}_{block} + \mathcal{S}_{control} + \mathcal{Q}_{controlledDribble}) \tag{18}$$

$$\mathcal{S}_{offense} = \frac{1}{2}(\mathcal{Q}_{kick} + \mathcal{Q}_{moveKick}) \tag{19}$$

The maximum velocities are then adapted in direct response to the currently observed opponent's offensive and defensive skill level, i.e. $f_{move} = \mathcal{S}_{offense}$ and $f_{turn} = \mathcal{S}_{defense}$.

Of course, the velocity factors could also simply be established depending on the current game score. However, the score might not always be available and the goal of this work is not to balance the game score but to adapt the difficulty of scoring a goal to an appropriate level for the human player.

5 Results

For the evaluation of our approach, we recorded twelve log files during games on both KiRo and StarKick. Three different human players – a beginner, an amateur and an advanced human player – played two games on each table. In one game, the robot played with maximum move and turn velocities. In the other game, the robot played very slowly so that the human players had more opportunities to play "their style" without being disturbed. Each log file contains the raw video data recorded at 50 Hz, summing up to a total amount of 14.4 GB of data for evaluation.

For evaluating the action classification, we hand-labeled the log files specifying at which positions the system should assess a certain action. However, as stepping through a log file is a very time consuming process, we only evaluated the first minute (3000 frames) of each log file. Figure 8 shows the results of comparing the manually created *ground truth* to the output of our system for KiRo (a) and StarKick (b). Always the human opponent and the robot were evaluated. We distinguished between *correct* classifications, *mismatches*, *false positives* and *false negatives*. When a different action than stated in the ground truth was recognized, a mismatch was counted. When an observed action didn't appear in the ground truth, a false positive was recorded. Conversely, if the ground truth contained an action where no action was recognized, a false negative was counted.

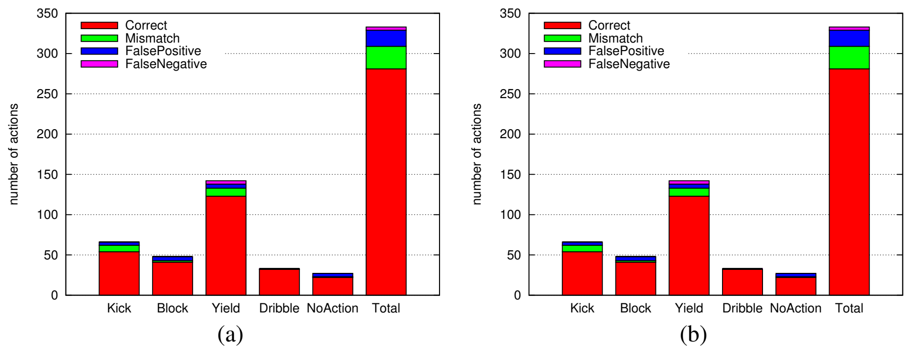


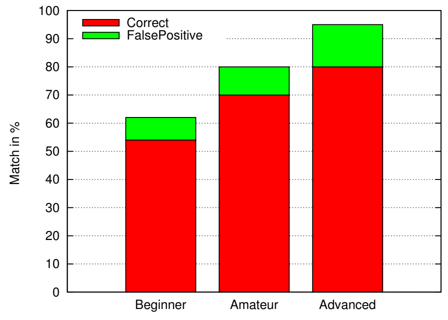
Fig. 8. Action classification results for games against KiRo (a) and games against StarKick (b).

[Discussion. . .]

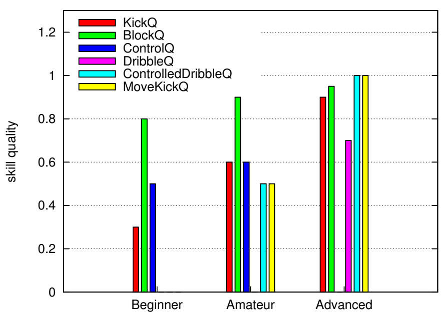
For evaluating the modeling of the opponent, we compared the classification results to the real level of expertise of the human players Figure 9(a) shows the classification results for each playing level. Figure 9(b) shows the quality measures averaged over the log files corresponding to the same human player. It can be seen, how the qualities increase with an increasing level of expertise.

[Discussion. . .]

For evaluating the behavior adaption, an advanced human player played a two minute game against StarKick. In the first minute, the human played like a beginner. Then, it played the best as possible. Figure 10(a) shows the velocities assessed according to the currently determined opponent playing level and Figure 10(b) depicts the velocities with respect to the currently assessed defensive and offensive playing skills. During the game, StarKick uses the velocities of Figure 10(b) for adapting to the opponent.Starting with maximum velocities it decreased its velocities in order to meet the



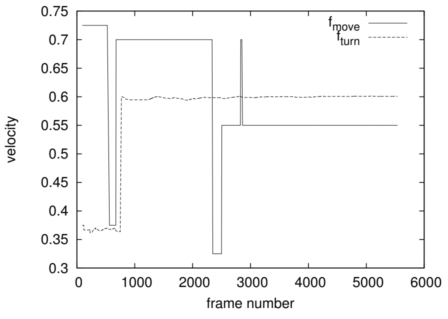
(a)



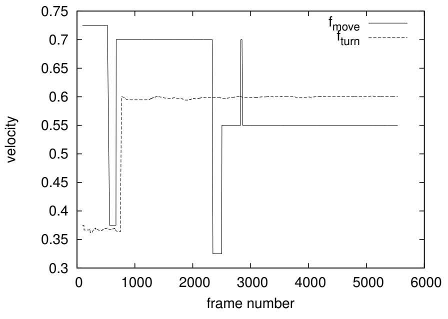
(b)

Fig. 9. Classification results for the human players with respect to their level of expertise (a) and assessed skill qualities with respect to the human player's level of expertise (b).

human's playing level. After the human player increased its performance after a minute, also StarKick started to raise its game velocities again.



(a)



(b)

Fig. 10. Evolution over time of the adapted move and turn velocities (a) (b).

[Discussion. . .]

6 Conclusion

In this paper, we presented an approach for automatically adapting the behavior of an autonomous table soccer robot to a human opponent. Experiments show, that it is possible to recognize basic actions in a robust way, that the opponent can reliably be classified with respect to different levels of playing expertise, and that the robot adapts its behavior successfully to the playing level of the opponent player.

In the future, we would like to adapt not only the move and shoot velocities but also respond with alternative actions to the opponent's particular playing characteristics. For an even better understanding of the opponent behavior, additionally its playing style could be assessed and general observations about the course of the game, e.g. the ball's distribution over the field could be taken into account.

References

1. Sutton, R., Barto, A., eds.: Reinforcement Learning: an Introduction. MIT-Press, Cambridge, Massachusetts (1998)
2. Weigel, T., Nebel, B.: KiRo – An Autonomous Table Soccer Player. In Kaminka, G., Lima, P., Rojas, R., eds.: RoboCup-2002: Robot Soccer World Cup VI. Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York (2003) 119–127
3. Weigel, T.: KiRo – A Table Soccer Robot Ready for Market. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2005)
4. Han, K., Veloso, M.: Automated Robot Behavior Recognition Applied to Robotic Soccer. In Hollerbach, J., Koditschek, D., eds.: Robotics Research: The Ninth International Symposium. Springer-Verlag, London (2000) 199–204
5. Riley, P., Veloso, M.: On Behavior Classification in Adversarial Environments. In Parker, L., Bekey, G., Barhen, J., eds.: Distributed Autonomous Robotic Systems 4. Springer-Verlag, Berlin, Heidelberg, New York (2000) 371–380
6. Riley, P., Veloso, M.: Planning for Distributed Execution through Use of Probabilistic Opponent Models. In: Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS). (2002) 72–81
7. Weigel, T., Zhang, D., Rechert, K., Nebel, B.: Adaptive Vision for Playing Table Soccer. In: Proceedings of the 27th German Conference on Artificial Intelligence, Ulm, Germany (2004) 424–438