

Interpretation and Processing of Position Data for the Empirical Study of the Behavior of Simulation League RoboCup Teams

Michael Beetz, Bernhard Kirchlechner, and Felix Fischer

Informatik IX, Technische Universität München
D-85748 Garching bei München, Germany
{beetz|kirchlec|fischerf}@in.tum.de

Abstract. One of the key problems in the study of multiagent systems in which the agents exhibit continuous behavior is the automatic recognition and analysis of intentional activities based on observable behavior. Such an automated analysis requires software systems to structure continuous motions into episodes that are meaningful in the respective application domain, to acquire and maintain models of agent activities, and to use such models to reason about multiagent system behavior. In this paper we describe the application of automatic interpretation methods to the empirical study of the playing behavior of teams in the RoboCup simulation league based on motion and episode models.

1 Overview

Football games considered as multiagent systems are excellent example applications for the automated recognition and analysis of agent activities based on observable behavior. The automated analysis of football games requires integrated computer and sensor systems that are capable of (1) acquiring position and motion data of the players and the ball, (2) interpreting these data in terms of intentional activities, such as dribblings and passes, and (3) assessing the game based on an abstract game model that is obtained through the interpretation of the data.

The research reported in this paper is part of an ambitious, mid-term project that studies the automated analysis of football games. The main objectives of the project are (1) the investigation of novel computational mechanisms that enable computer systems to recognize intentional activities, (2) the development of an integrated software system to automate game interpretation and analysis, and (3) the demonstration of the impact of automated game analysis on application areas, such as sport science, football coaching, and sports entertainment. The results will be showcased in the form of an intelligent information system for the games at the Football World Championship 2006 in Germany.

The physical setup that is currently developed by Cairos Technologies AG and the Fraunhofer IIS consists of tiny microwave senders placed into the ball and the shin guards of football players. These senders broadcast signals with

high frequency (the ball 2000 and the shin guards 700 times per second). Eight antennas that are placed at optimal positions within the stadium receive the signals. A position estimation system determines the position of each microwave sender with an expected accuracy of 5-10cm.



Fig. 1. Microwave-based position tracking system: (a) receiver; (b)&(c) installation; (d) sender.

A first prototype for one player and the ball has been demonstrated in April 2003. The system will be fully operational for two complete teams and the competition ball in Fall 2004. Because we cannot get real data until the end of summer 2004 we currently develop our motion and episode acquisition and recognition methods for data from the RoboCup simulation league. This is where the application data and examples in this paper come from.

1.1 Key Ideas of the Game Analysis

There are two important ideas that guide the design and implementation of our game analysis system: first, the acquisition, maintenance, and use of abstract game models and second, the idea of acquiring the model by inferring the one that is most likely given the sensor data.

Key Idea 1: Model-based Game Analysis The first key idea is to perform football game analysis by first acquiring and maintaining a model of the happenings of the game and then using the model to solve the relevant reasoning tasks. A model is a simplified description or an abstraction of a system, used to reason about the system, to explain how something works or to calculate what might happen.

According to Perl [8, 2] and Lames [4] model building is the mapping of the real game process into an abstract representation formulated in features relevant for game analysis. The model is designed w.r.t. the goals of game analysis.

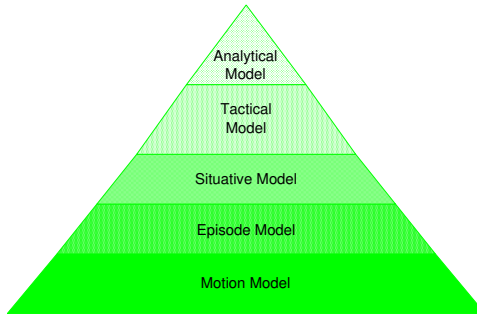


Fig. 2. Pyramid model of football analysis.

Our model used for game analysis is a layered one and depicted in figure 2. The most basic layer of the model is the *motion model*. It represents the motions of players and the ball in a compact way that facilitates the recognition of game episodes. The *episode model* segments the continuous motion model into episodes relevant for game analysis. Examples of such episodes are dribblings, passes, shots, offensive plays, etc. Episodes are characterized by a starting and ending time and an episode class that describes what is going on in the episode. The *situative model*, in which the impact of players on the episodes is described comprises processes, interactions, and the role of players. The *tactical model* that represents the objectives and classifies the tactical actions of the players, comprises player specific intentions and possible play actions like pass or shot opportunities. Finally, the *analytical model* represents causal relations and analytical results.

Key Idea 2: Probabilistic Estimation of Models The second key idea is to pose the acquisition of game models as a probabilistic reasoning problem. Even for the RoboCup simulation league, where the game state is fully observable, is it not possible to perform the episode recognition correctly and accurately. Even if we knew that a particular simulated robot performs a dribbling control routine this does not mean that this player also exhibits a dribbling behavior. Non-deterministic action execution and interference with other players often yields strong variations in the players' behavior. The situation gets even worse when we move from simulated games to real ones where the game situation is only partly observable. Because we only have position sensors in the ball and in front of the players' shin, there are many situations in which we cannot decide which player has played the ball. In addition, since we perform real perception in the real world the sensor data received are sometimes unreliable and inaccurate.

To deal with these issues we pose the model acquisition as a probabilistic inference problem. In this setting we want to determine the model m^* , which is the most likely one of all models m given the sensor data d , that is $m^* = \operatorname{argmax}_m P(m \mid d)$. For the remainder of the paper we will assume that the

motion model can be acquired accurately, while the episode model must be determined using probabilistic reasoning methods.

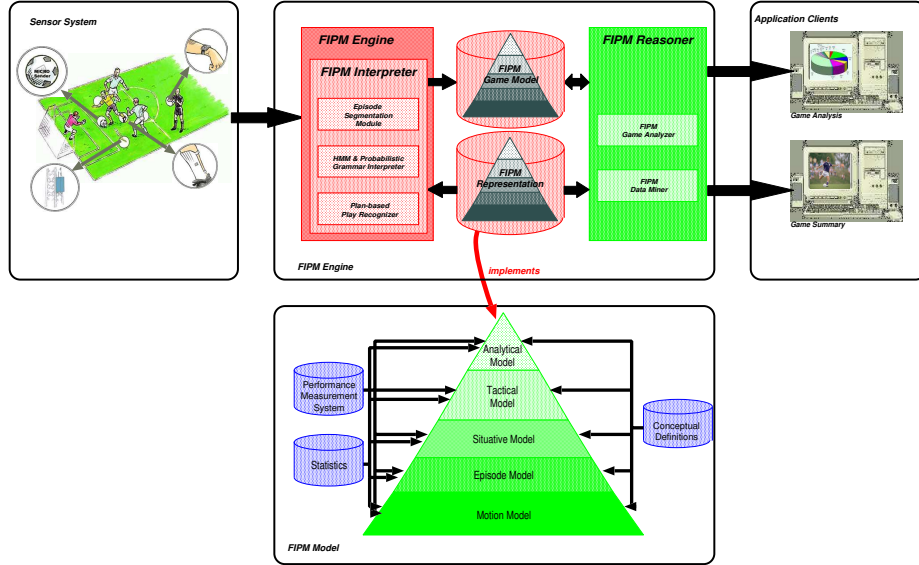


Fig. 3. Software architecture of the system.

1.2 System Overview

Sensor data interpretation and game model acquisition is performed in a sequence of four steps. First, the state of the game and the motion of the ball and all players are estimated probabilistically to account for missing data and inaccurate measurements. The result is a complete motion and event model of the game. In the second step, the motion data is segmented into meaningful episodes, and these are then classified as passes, dribblings, etc.

The main system components that are relevant w.r.t. this paper are the *motion interpreter*, the *episode interpreter*, and the *database* that stores the motion and the episode model. The motion interpreter takes the raw position data (which is taken from log files in the case of data from the RoboCup simulation league) as its input and computes a compact motion model of the evolution of the game. The episode interpreter uses the motion model and additional event data to recognize and analyze meaningful game episodes.

2 The Game and its Models

In this section we will introduce the notion of a game and describe how games can be represented at a level of abstraction appropriate for our purpose. Later in sections 3 and 4 we will use these abstract game models in order to perform our empirical analyses.

2.1 The Game Data

Intuitively, we consider a game to consists of a sequence of positions for each player and the ball and some additional instantaneous events such as offsides, corner kicks, throw-ins, goals etc.

Thus, we formalize a *game* as a pair $\langle \text{Pos}, \text{Evs}_{\text{obs}} \rangle$ where $\text{Pos} : \mathbb{T} \times \mathbb{O} \rightarrow \text{COORD}$ is a function that maps any time instant t during the game and any object o on the field to the position of o at t . Further, we introduce Evs_{obs} as a partially defined mapping of time instants to the event types that denote decisions of the referee such as “goal”, “corner kick”, “free kick”, etc.

For our subsequent discussion it is more convenient to formalize a game in first order logic. To do so, we introduce the following predicates:

- **pos** (o, p, t_i) that is true if and only if $\text{Pos}(o, t_i) = p$.
- **occurs** (ev, int) that is true if and only if the event ev occurs in the time interval int , or more precisely if ev starts at the begin of int and ends at the end of int . Note, we consider time instants to be time intervals with the duration 0.
- **event-type** $(ev, e\text{-type})$ that holds if $e\text{-type}$ is the type of event ev . At the game data level the set of event types is restricted to $\{\text{BEFORE-KICK-OFF}, \text{TIME-OVER}, \text{PLAY-ON}, \text{KICK-OFF}_i, \text{KICK-IN}_i, \text{EVENT-FREE-KICK}_i, \text{CORNER-KICK}_i, \text{GOAL-KICK}_i, \text{EVENT-GOAL}_i, \text{DROP-BALL}, \text{OFFSIDE}_i, \text{PENALTY-KICK}_i, \text{FIRST-HALF-OVER}, \text{PAUSE}, \text{FOUL-CHARGE}_i, \text{FOUL-PUSH}_i, \text{FOUL-MULTIPLE-ATTACK}_i, \text{FOUL-BALLOUT}_i, \text{BACK-PASS}_i, \text{FREE-KICK-FAULTS}_i\}$.

In this formalization i indicates the team and can have the values *left* or *right* depending on the team playing from left to right or right to left, respectively. Using these predicates we can completely represent a game as a set of facts.

2.2 The Abstract Game Representation

While the above representation resembles a complete model of a game, it suffers from two main problems, namely that it is not compact and that it has no structure that could be exploited by inference mechanisms to cut down the search space. To mitigate these problems we will now introduce a more abstract representation of games.

One of the main differences between the raw game data and the abstract representation of games are the types of events used to model their evolution. We will start with the basic event types *motion-segment* (o, p, f) and *ball-contact* (p) .

The purpose of the motion segment event types is to structure the motions of the individual objects into motion segments that can be described uniformly. Thus, a motion model for object o is a sequence of motion segments $m = m_1, m_2, \dots, m_n$ such that the endpoint of segment m_i is the starting point of segment m_{i+1} and the end time of the motion segment m_i is the starting time of segment m_{i+1} . The trajectory of an object a within a given motion segment is represented as a piecewise function f that maps time indices to the respective x and y coordinates of the object. The function f is defined for all t with $t_1 \leq t \leq t_2$

and $f(t)$ returns the position of o at t . The individual tuples m_i have the form $\langle o, t_1, t_2, p_1, p_2, f : \mathbb{T} \rightarrow \mathbb{P} \rangle$ where o denotes the object (ball or player), t_1, t_2 the start and end time of the motion segment, and p_1, p_2 the respective start and end position. To assert a motion segment $m_i = \langle o, t_1, t_2, p_1, p_2, f : \mathbb{T} \rightarrow \mathbb{P} \rangle$ as a fact in our motion model we formalize it as $occurs(motion-segment(o, f), [t_1, t_2])$.

In addition to that, we extend the set of instantaneous events with the event type $ball-contact(pl)$. Thus, we state a ball contact $ball-contact(pl)$ at time instant t_i as $\exists ev \textbf{occurs}(ev, [t_i, t_i]) \wedge \textbf{event-type}(ev, ball-contact(pl))$.

The abstract model further structures the evolution of football games into ball actions. We want to distinguish four different classes of ball actions: (1) keeping control over the ball, (2) passing, (3) shooting, and (4) loss of ball possession. If ball actions had very high success rates we could recognize them using simple rules. A *ball possession* is a sequence of ball contacts of the same player. A *pass* is a ball contact of one player followed by a ball contact of another player (hopefully a team mate). A *shot* is a ball contact of a player followed by an out of bounds event, where the out of bounds event occurs in or close to the goal.

A player performs an action of a given action type if it kicks the ball with the intention to perform that action. Thus, the performance of an action does not pose any constraints on the change of the world state.

Hence, the abstract model does not store facts for the *pos* relation explicitly. Rather, we infer the position of object o at time instant t from the respective motion segment. Additional predicates required for our analysis tasks will be introduced as needed.

3 Analyzing Shots, Shot Opportunities, and Their Preparation

Let us consider an analysis task that we perform to better understand the performance of teams in simulated robot soccer. An aspect of team performance that has direct impact on the game outcome is the preparation of shot opportunities, the shooting skill, and the criteria applied to decide whether or not to shoot in a given situation. To identify the key factors, we investigate and compare the preparation and usage of shots by successful and unsuccessful soccer teams in the middle and final round of the simulation league competition of the 2003 RoboCup event.

To start our empirical analysis we have to pin down the notions in our analysis task precisely. In particular, we have to specify what we mean by successful and unsuccessful soccer teams and how we define scoring opportunities. Clearly, the results of our analysis strongly depend on the precise specification of these notions.

Various alternatives exist in order to define “successful”. For example, we can define it using the overall score of a team, such that a team is successful if the sum of scores obtained in each individual game is in the top quartile of all teams. Figure 5 shows how this notion of success can be implemented as a

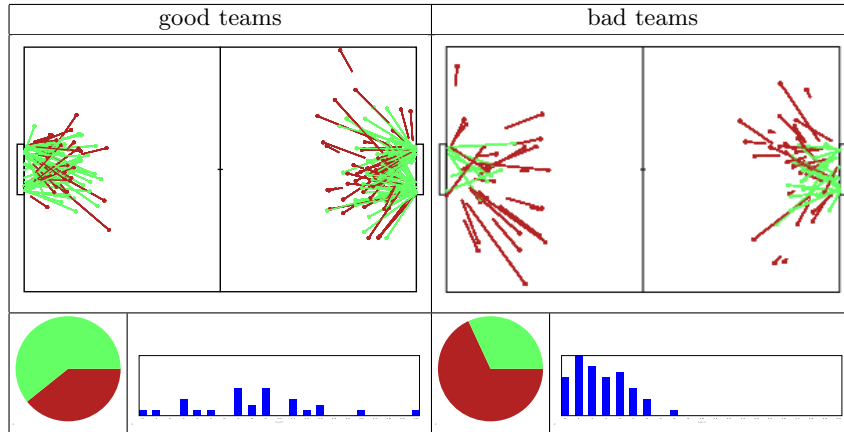


Fig. 4. The set of successful and unsuccessful shots of the successful and unsuccessful teams and additional descriptive statistics.

query on the motion model. Another possibility is to take the top n teams of the tournament.

```

select team from <teams with score-difference  $score\_d$  for each game>
group by team
having sum((score.d $\geq$ 0) + (score.d $\geq$ 1) + (score.d $\geq$ 3)) $\geq$ <score threshold>

```

Fig. 5. SQL query that computes the set of successful teams. For each game, a team receives one, two or three points for a tie, a close or a high win, respectively. Teams are then selected based on their overall score and a threshold (which itself may be computed using a similar query to refer to the top quartile w.r.t. the set of all teams).

More interesting is the concept of scoring opportunity because it is much harder to define. Experts usually use their judgment to decide. Thus we give a more intuitive definition. A scoring opportunity is a game situation in which the players typically decide to shoot. In these situations the players themselves believe that they can score. In addition, we weigh these situations with the likelihood that they will score from these situations. We then apply decision tree learning in order to learn the precise definition from the previous games. This bears the additional advantage that the notion of scoring opportunity is tailored to the performance level of the competition.

3.1 Assessing the Level of Shooting Skills

Let us now have a closer look at the skill level with which the successful and unsuccessful teams perform the shots. We define the skill level as the probability distribution over the shot outcomes conditioned on the type of situation.

This raises two issues. First what kind of abstraction do we want to use in order to characterize the situations in which the players shot the ball. Second, how do we find types of situations that are highly correlated with the outcome

of shots. Clearly, the highly correlated features would result in more meaningful explanations for the shot outcome. Our solution to this problem is to define a feature language for situations in which the players tend to shoot the ball. We then use decision tree learning in order to obtain probabilistic classification rules for predicting the outcome of shots given the set of shots of the good versus the bad teams.

The feature language that we use comprises the following features: *distance* (distance between ball and the goal at the kicking moment), *defenders* (the number of defenders closer to the goal), *dribbling length* (the minimum distance the attacker can carry the ball without being attacked) and *scoring-angle* (the largest angle segment in which a shot would hit the goal). The last feature already considers the segments of the goal that are blocked by defensive players.

We use the following mathematical definitions for these features. The *goal distance* is the distance from the ball position to the center of the goal at the time of the kick. As *defenders* we consider those opponents whose goal distance is smaller than that of the attacker. The *dribbling length* is determined by calculating the position at which each of the defenders could intercept the attacker. To determine the *shooting angle* we first determine the segments on the goal line that are not occluded by any opposing player (see figure 6). In a nutshell, the

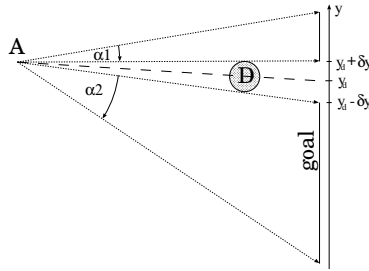


Fig. 6. Calculation of the shooting angle

interval in which the kicker can kick into the goal without the defenders being able to block the shot is $I = I_0 \setminus \bigcup_{d \in Defenders} [y_d - \delta y_d; y_d + \delta y_d]$ as shown in figure 6.

Using these features we learn a classification function that predicts the success of a (possible) shot. To do so we apply decision tree learning to the actual shot instances. The result of this learning process is a set of rules that mapping situation classes characterized using our feature language into the predicted outcome of a shot, where each rule has an associated expected accuracy.

The following rule predicts a successful shot for the good teams with an expected accuracy of 70%. It states that a shot will succeed in situations where the shooting angle is smaller than 160° and where at most one player (usually the goalie) is left to cover the goal:

if (*shootingAngle* ≤ 2.793 **AND** *defenders* ≤ 1)
then with probability 70% classify *successful* = *true*
supported by 60/166 samples

Even more informative than having a characterization of the shooting skills of the successful and unsuccessful teams is an analysis of the situations where these skills differ. This information would enable learning and advice taking robot soccer teams to improve their shooting skills in a directed way. For example, reinforcement learning football players of the unsuccessful teams could explore the situation space more carefully in order to find policies that have a higher expected utility. Or, a “trainer” agent could advise the players to use the shooting directions and strength of the successful players.

In order to obtain a model of the differences between the shooting skills of successful and unsuccessful teams, we use the decision trees that we have learned in the previous step (recall that these trees predict whether a shot in a given situation is successful or not for the successful and the unsuccessful teams, respectively). We then take a set of shooting situations, predict for every situation whether the successful and the unsuccessful teams will shoot successfully, and classify the situations into four categories: the one where both succeed, those where only the good ones or the bad ones succeed, respectively, and those where both fail. In our case, this for example led to the following rule describing situations in which the good teams succeed and the bad ones do not:

if (*defenders* ≤ 1 **AND** *dribblingLength* ≤ 5.46 **AND** *shootingAngle* ≤ 2.98)
then with probability 100% classify *predgood*=*true* **AND** *predbad*=*false*
supported by 49/213 samples

In order to use this analysis for coaching soccer teams it might be better to switch to a feature language which decomposes the feature *scoring angle* into a set of component features. Such a feature language would allow us to distinguish between situations in which the player has an inappropriate angle to the goal from other ones where many defensive player block the way to the goal.

3.2 Inferring Preference Models for Shots

We will now infer a model for the situations in which the teams tend to shoot the ball and those in which they do not. Again we will learn the classification of the respective situation categories using decision tree learning. To do so, we need an adequately distributed set of training instances.

To extract this set and analyze the situations we introduce the following additional features: the *pass safety* (1-probability that the pass fails), the *shooting value* (a measure for the probability to score a goal), and the *total value* of an attacker. The *pass safety* is estimated by considering the angle in which no defender can intercept the ball, the *shooting value* is a combination of the results of section 3.1 and the *total value* is calculated as the product of the former two features. The training examples can then be obtained by means of a threshold on the attackers’ shooting value. The particular features used for classification are *maxOther* (the maximum total value of the other attackers) and *avgThreeBest* (the average total value of the three best attackers).

The result of these learning task are predictive classification rules such as the following:

```
if (shootingAngle ≤ 1.96 AND goalDistance > 13.2)
  then with probability 93% classify shot=false
  supported by 426/533 samples
```

Again we can get more informative models of the action preference models by performing multi-step analyses, comparing the situations in which the bad teams shoot and the good ones don't and vice versa. A kind of analysis which is particularly interesting from the point of view of the bad teams is to infer classes of situations in which they decide not to shoot, but where the predicted success probability is high. If this is the case, as suggested by the following rule, changes to the action preference model of the respective team promise to result in a higher expected utility.

```
if (dribblingLength > 5.4 AND defenders ≤ 1)
  then with probability 87% classify shot=false AND success=true
  supported by 8/80 samples
```

3.3 Preparation of Shot Opportunities

Of course, one problem might be that a team does not create enough shot opportunities. In order to analyze this aspect of the game we can look at the episodes in the shot preparation phase of the game. To do so we can analyze the pass selection. That is, we investigate whether there have been passing options in the preparation phase and assess the risk of failing the pass to that player.

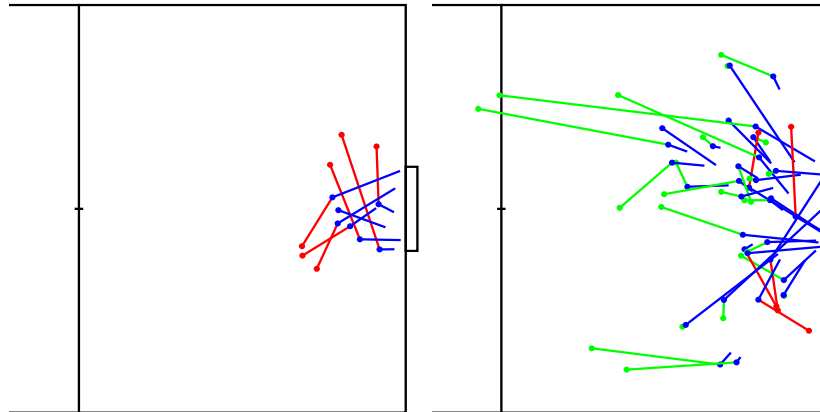


Fig. 7. The shots and their preceding episodes for a good team and several bad teams

Comparing the preparation of shots for good and bad teams we see (figure 7) that teams with good performance tend to play passes into the area in front of the goal whereas the bad teams in the majority of cases dribble a long way before shooting at the goal. Additional examinations will give us a deeper insight in the correlation of ball action preference and performance of the different teams.

3.4 The Impact of Shot Performance on Game Results

So far, we have simply assumed that the above aspects of shot selection, performance, and preparation have a big impact on the results of the teams. We can check whether such an assumption is reasonable by learning a predictive model of the outcome of games based on the aspects of shot performance that we have inferred above. Cross-checking this model then allows us to confirm the assumption.

Having such a model also enables us to examine what abilities are the more important ones. Thus we will be able to find the skills teams should focus on with their training efforts.

4 Using the Game Model

This section describes some examples of empirical studies that use the described motion and episode models to answer relevant queries. The section demonstrates the breadth of analysis queries that we can answer based on our abstract game model. The section starts with some simple queries on the models for assessing their accuracy, compression rate, and the computational resources needed, then it gives an example of a set of features derived from them, study scoring chances and the preference model of actions.

4.1 Querying the Game Model

One of the main criteria for assessing the motion and episode models is the degree to which they support the automatic analysis of football games. In this section we will show that by using two characteristic classes of queries: first, queries about the motions of the ball and the players and second, queries about the passing game within a team.

A typical abstract motion query is about the distance that every player was moving during a game episode. This can be retrieved through a simple pseudo SQL query:

```
select object, sum( $\sqrt{(x_{end} - x)^2 + (y_{end} - y)^2}$ ) from motion  
where object between 0 and 10 group by object
```

Note that answering the query is computationally efficient because the starting and endpoints of motions are explicitly stored in the motion tuples. The SQL query has only to compute the sum of the distances of the individual motion segments of the players. Analogously, we can query the velocity of passes to draw inferences about the playing characteristics of players and teams by inferring the average velocity of passes, or the distribution of passes, which can be inferred from the length of passes that were played. Much of the information retrieved in these example queries is already cached in the motion model, which makes these queries to be answered very fast.

The passes played by the individual players can also be easily retrieved from the model and visualized (Figure 8(a)):

```

select player, count(*)
from episodes join episodes_pass using(id)
group by player

```

The query is even more informative if we group the passes by their players and receivers:

```

select player, player2, count(*)
from episodes join episodes_pass using(id)
group by player1, player2

```

The result can be visualized as a directed graph (figure 8(b)) where the nodes are players and the thickness of arcs indicates the number of passes played between them. We can also query the fraction of successful and unsuccessful passes and shots and display them as piecharts.

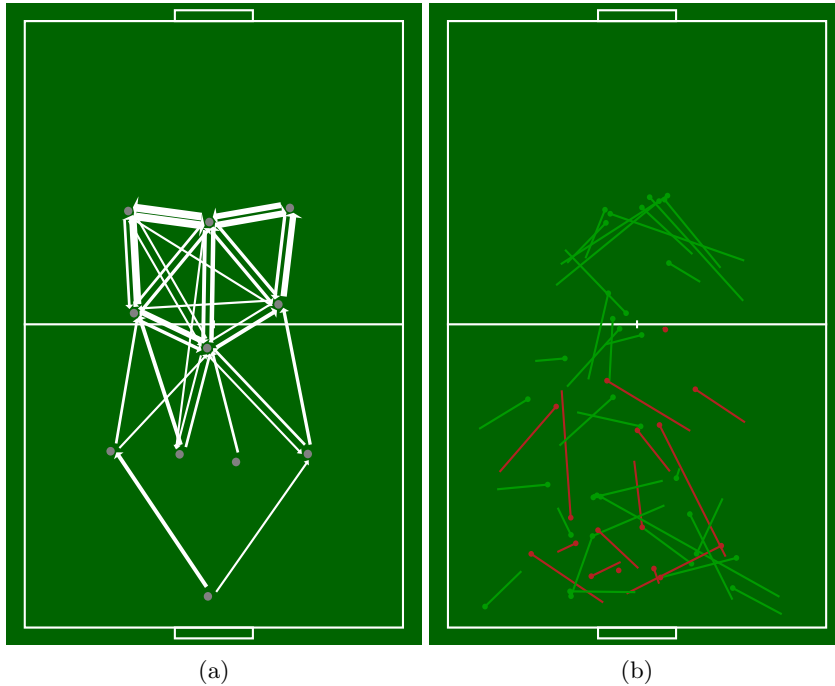


Fig. 8. (a) passing channels and (b) passes for selected players

4.2 Complex Queries

Nair et. al. [7] propose an Individual Agent Model for RoboCup soccer consisting of eight individual features. As an example for queries with higher complexity, we will show how these features can be computed from the motion and episode models.

For reasons of clarity, we use a temporary table `episodes_pos` containing the positions for each player at the start of each episode. While the motion and

episode models might not explicitly contain this information for players not directly involved in the respective episode, it can efficiently be computed from these. We further use a coordinate system with the origin located at the center of the field and the first team playing in the direction of the positive x axis and refer to the maximum values for x and y (i. e. half the width and height of the field) by means of global variables $@xm$ and $@ym$.

This said, the individual features are given as follows:

- Ball Velocity
select velocity_start **from** episodes **join** episodes_shot **using**(id)
- Distance to Goal
select $\sqrt{(@xm * \text{sign}(10.5 - \text{player}) - x)^2 + y^2}$
from episodes **join** episodes_shot **using**(id)
- Extrapolated Goal Line Position
select $((@xm * \text{sign}(x_{\text{end}} - x) - x) * (y_{\text{end}} - y) / (x_{\text{end}} - x) + y)$
from episodes **join** episodes_shot **using**(id)
- Angle from Center of Field
select **degrees**(**atan**($y / (@xm * \text{sign}(10.5 - \text{player}) - x)$))
from episodes **join** episodes_shot **using**(id)

For features that involve additional players, we use an additional (temporary) table `defender_dist` listing for each shot the distances between the shooter and each player (but the goalie) of the opposing team, which can efficiently be generated from motion and episode models using the query

```
select id, player, object as other, xo, yo,  $\sqrt{(x - xo)^2 + (y - yo)^2}$  as dist
from episodes join episodes_shot using(id) join episodes_pos using(time)
where (other between 1 and 10 or other between 12 and 21) and
(other > 10) != (player > 10) .
```

With this, the remaining features are given as follows:

- Number of Defenders
select **sum**(**degrees**(**atan**(($y - yo$) / ($x - xo$)))) **between**
degrees(**atan**(($y + @ym$) / ($@xm * \text{sign}(10.5 - \text{player}) - x$)))) **and**
degrees(**atan**(($y - @ym$) / ($@xm * \text{sign}(10.5 - \text{player}) - x$))))
from (episodes **join** episodes_shot **using**(id))
left join defender_dist **using**(id, player)
group by id, player
- Distance of Closest Defender
select min(dist)
from episodes **join** episodes_shot **using** id **left join** defender_dist **using**(id)
- Angle of Closest Defender w.r.t. Goal
select **degrees**(**atan**($yo / (@xm * \text{sign}(10.5 - \text{player}) - xo)$))
from (episodes **join** episodes_shot **using**(id)) **left join** defender_dist **using**(id)
- Angle of Defender from Shot
select **degrees**(**atan**(($yo - @aim$) / ($@xm * \text{sign}(x_{\text{end}} - x) - xo$)) -
atan * (($y - @aim$) / ($@xm * \text{sign}(x_{\text{end}} - x) - x$))))
from (episodes **join** episodes_shot **using**(id)) **left join** defender_dist **using**(id)
where $@aim$ denotes the extrapolated goal line position as computed above

The results of these queries can then be used to learn rules for successful and failed shots, to detect scoring opportunities, etc.

5 Related Work

Our research is closely related to that of Intille and Bobick [3, 1] who analyzed offensive plays in American football through script-like recognition mechanisms. While plays in American football are usually well planned and the actions do not have such nondeterministic outcomes as football actions, script-based recognition methods would not work that well for football. Visser and Weland [10, 6] like Riley and Veloso [9] propose interesting approaches to online learning of decision rules for opponent behavior. They address, however, not the problem of constructing such comprehensive activity models as we do in our project.

The problem of acquiring motion models for sport games has been addressed by Little and Gu [5]. Since they acquire trajectory data for ice hockey games the trajectories are much better described by simple curves.

6 Conclusions

In this paper we have described some applications studying empirically the behavior of RoboCup football teams. These applications use a layered model hierarchy for football games acquired from positional data. We have argued that the objectives for these models are to provide reliable information about ball contacts of players and classification of ball actions and definitions for successful actions. We have shown that these models are expressive enough to answer essential queries in game analysis. We have illustrated four examples using these models: Simple database queries to get statistical data for individual players or teams, a system of features describing the behavior of teams, a study of scoring chances for an analysis of the performance of a team and an action preference model contrasting the behavior of different teams in different situations.

In our ongoing research we use the models to acquire abstract models of more complex activities such as offensive plays, we also acquire and use them in conjunction with the described applications to interpret and analyze games of the simulation league world championship. Our ultimate goal is to develop a comprehensive model-based game analysis system that works on real position data and is deployed at the soccer world championship 2006 in Germany.

References

1. A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing, 1998.
2. T. Hein and J. Perl. Einsatz komplexer informationssysteme in der sportspielforschung. In G. Hagedorn and N. Heymen, editors, *Methodologie der Sportspielforschung*, pages 141–149. Czwalina, Ahrensburg, 1992.
3. S. Intille and A. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks, 1998.
4. M. Lames. Probleme von beobachtungssystemen in den sportspielen am beispiel fußball. In *Analyse und Beobachtung in Training und Wettkampf*, dvs Band 47. Verlag Academia, Sankt Augustin, 1992.

5. J. Little and Z. Gu. Video retrieval by spatial and temporal structure of trajectories. *SPIE Storage and Retrieval for Media Databases*, 2001.
6. A. Miene, U. Visser, and O. Herzog. Recognition and prediction of motion situations based on a qualitative motion description. In *RoboCup 2003*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2003.
7. R. Nair, M. Tambe, S. Marsella, and T. Raines. Automated assistants to analyze team behavior. *Journal of Automated Agents and Multi-agent Systems (JAAMAS)*, 2004.
8. J. Perl. Möglichkeiten und grenzen der erfassung von sportspielhandlungen. In *Computer- und Medieneinsatz im Fußball*, volume dvs Band 109. Czwalina Verlag, Hamburg, 2000.
9. Patrick Riley and Manuela Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.
10. U. Visser and H.-G. Weland. Using online learning to analyze the opponent's behavior. In *RoboCup 2002*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2002.