Camera-based Observation of Football Games for Analyzing Multi-agent Activities

Michael Beetz
beetz@in.tum.de

Jan Bandouch
bandouch@in.tum.de

Suat Gedikli
gedikli@in.tum.de

Nico v. Hoyningen-Huene
hoyninge@in.tum.de

Bernhard Kirchlechner
kirchlec@in.tum.de

Alexis Maldonado
maldonad@in.tum.de

Intelligent Autonomous Systems Group
Technische Universität München, Munich, Germany

ABSTRACT

This paper describes a camera-based observation system for football games that is used for the automatic analysis of football games and reasoning about multi-agent activity. The observation system runs on video streams produced by cameras set up for TV broadcasting. The observation system achieves reliability and accuracy through various mechanisms for adaptation, probabilistic estimation, and exploiting domain constraints. It represents motions compactly and segments them into classified ball actions.

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*Video analysis*

General Terms

Motion tracking, Analysis of intentional activity

Keywords

video analysis, state estimation, object tracking, motion interpretation

1. INTRODUCTION

In order to realize computer systems that can interpret and analyze cooperative and competitive activity in multi agent systems, we need to realize powerful observation systems for agent behavior. One domain that is intensively studied and has proven to be a very interesting and challenging domain in the multi-agent community is football — in particular in the context of the RoboCup competitions.

In this paper, we investigate the observation of real football games based on video streams provided by a set of TV cameras. This computational problem is interesting because it can be solved with an ordinary setup for broadcasting football games without further additions. Based on the camera image streams the software system computes a representation of the game that enables it to answer questions such as: What are the characteristic offensive plays of the two teams? What are the strengths/weaknesses of a particular team/player? What roles do the players have? Do their capabilities match their roles? Do they achieve their tasks? How does a team create scoring opportunities? What are each players' skills? What is the tactical formation of a team?

Here we consider the problem of generating an action model suitable for answering such questions. The knowledge representation mechanisms needed for inferring such answers and the application of these techniques to the analysis of football games are investigated in a companion paper [3].

Building up a game representation for analyzing multi-agent activity requires a software system to exhibit the following capabilities,

- The reliable and accurate estimation of the positions of the players and the ball.
- The compact representation of motions and their segmentation into actions

The realization of these capabilities is also very difficult for various reasons. The positions, direction, and zooming factor of the camera are not known and must therefore, be estimated from the image stream. Processing the image streams is made difficult by the change in lighting conditions. These conditions change substantially when taking a camera sweep from one part of the field to another one or from one moment to the next when clouds are passing. Other complications include the inaccuracy in depth estimations caused by the low height of the camera positions and the fact that TV cameras are typically placed only on one side of the field resulting in players at the other side being very small in the image. There are also frequent occlusions of players by other ones.

The contributions of this paper are twofold. First, we describe the design, implementation, and empirical analysis of a camera-based observation system for football games. Second, we demonstrate how abstract representations of the game actions can be inferred that enable automatic game analysis systems to represent and reason about the actions in multi agent systems in sophisticated ways. These techniques are described in a companion paper [3].

In the remainder of the paper we proceed as follows. The next section presents an overview of the observation and interpretation mechanisms of the system. Section 3 details the components and methods used for the visual perception of the players and the ball. Section 4 then describes the probabilistic estimation methods that

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

enable the system to keep track of players on the field and estimate their postions more accurately. The interpretation of position data in terms of actions is detailed in section 4. We conclude with a discussion of related work and our conclusions.

2. SYSTEM OVERVIEW

Before we dive into the technicalities of the vision-based game analysis system let us first describe the sensing apparatus of the system and the overall software architecture.

2.1 Physical Setup

The physical setup of the system consists of a set of cameras used to broadcast football games on television. The cameras are pointed and zoomed by a camera man and provide complete image streams. The game observation system passively receives the camera stream without having control over it, and without receiving information about the pan and tilt angles of the camera nor its zoom parameters.

The cameras do only partly cover the playing field and not all players are visible all the time. In addition the cameras usually focus on the surroundings of the ball that is information about other parts of the field are only available at a substantially lower resolution. Another complication is the lack of height of the camera positions. Because the cameras are fixed in a maximal height of about 18 meters the depth resolution of a pixel imaging the opposite side of the field is about 1,5 meters. The sizes of players are only about 35 pixels where the course grained resolution of pixels blurrs the colors. Therefore the appearance of players close to the camera differs substantially from those of distant players. Finally, drastical and abrupt changes of lighting conditions complicate the reliable interpretation of color information.

2.2 Software Architecture

The software architecture of the game analysis system is depicted in figure 1. The system is decomposed into two main components: the model acquisition and the reasoning and model mining component.

The model acquisition component consists of the *visual perception module*, the *blob tracker*, and the *motion and action interpreter*. The *visual perception module* receives multiple streams of camera images and computes for each image the blobs that correspond to players, the referee, and the ball. A blob description contains an $\langle x, y \rangle$ coordinate on the field, a class label that specifies the team the player belongs to or whether it is a referee, and the covariance as a measure of the expected inaccuracy of the estimated position.

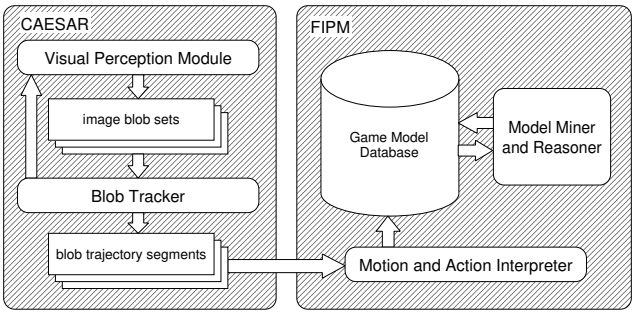

Figure 1: Software architecture of the FIPM analysis system.

The *blob tracker* integrates the observations generated by the *visual perception module* and produces trajectories of the observed object motions and associates these trajectories with the respective players. Using temporal redundancy and coherence the *blob tracker* estimates the player motions much more reliably and accurately. Additional computational tasks that the blob tracker performs are the association of player track identities with blobs and the completion of partly unobserved object tracks through probabilistic guessing.

The final component of the model acquisition component is the *motion and action interpreter*. The interpreter stores the motion data very compactly as a concise motion model and structures the continuous motions into application specific ball actions such as passes, dribblings, and shots. The interpreted motions and ball actions are then stored into the game model database that is used by the *reasoner and model miner* for game analysis purposes.

The *reasoner and model miner* learns situation specific models of the skills and action selection criteria of the football players using the game model database as the relevant set of experiences. It then uses the models in order to predict, analyze, and diagnose the teams' playing behavior.

The FIPM system is implemented as a distributed system running on multiple computers. In the current version the visual perception module for each camera is running on its own computer, so is the tracking, and the motion interpretation module. The modules communicate via Corba.

3. VISUAL PERCEPTION

As we have sketched before the computational task of visual perception is to estimate the 3D position of each player, the referees, and the ball in the camera view. A camera image and the recognized objects and their positions are shown in figure 6. Since the estimated positions are inaccurate the expected accuracy is computed and returned with the estimates.

Figure 2: Output of the visual perception. The players and the ball in the camera view are detected. The players are labelled with respect to the team they belong to. For each player, the referee, and the ball the $\langle x, y \rangle$ position is estimated.

The visual perception of the players and the ball based on a stream of camera images entails a number of difficult computational problems. The first problem is the camera position that gives the system a very flat viewing angle, which causes high inaccuracies in depth estimates. For example, if a player is at the opposite side of the field one pixel in the image corresponds to more than

1,50m depth. Also, if the camera does not zoom in on a player who stays at the other side of the field the size of the player in the image is about 35 pixels. This reduction in size also results in a considerable change of the visual appearance of players causing the color models of football dresses to exhibit much higher entropies. There is also frequent occlusion especially caused by one player covering another one. Luckily the space occluded by players is quite small because of the distance and position of the camera. Another issue is the rapid change of lighting conditions either due to change of cloudiness or due to the sharp shadow caused by the stadium roof in intense sun shine.

There are, on the other hand, also a number of assumptions that we can make in order to simplify the computational task of visual perception drastically [7]. The first assumption is the *ground plane assumption*. We can assume that all objects that interest us are standing on the field and have roughly the same size. Another important assumption is the *distinctive color* assumption: we assume that objects and their classes can be recognized by their colors. The football field is green and the lines white and the two teams are required to wear dresses that are visually easy to distinguish. We further know the static environment: the form of the field lines and all but a few parameters such as the length and the width of the field. Also since the frame rate is about 25 frames per second and the motions of the observing cameras and the players are not too fast, there is a high temporal coherence between subsequent camera images.

The visual perception of football games is decomposed into three closely interrelated subproblems: (1) the identification of the relevant color regions, (2) the estimation of the pan- and tilt angle of the camera and the zooming parameter, and (3) the identification and positioning of the players and the ball that are in the camera view. In the remainder of this section we will describe our approaches to the solution of these computational problems.

3.1 Color Segmentation

Since football is a spectator sport the playing field, the lines, and the dresses of the players are designed to be visually distinctive. From the visual features that allow for these distinctions color is the most informative one: the field is green, the lines are white, and the players of the teams are required to dress so as to achieve the highest possible contrast.

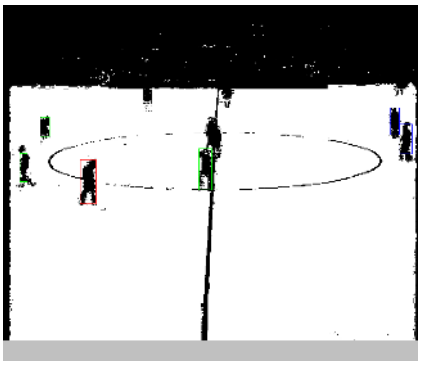

Figure 3: Computing the regions that are colored football field green.

Thus our first step in processing the camera images is to apply color classification and segmentation to the images. Given a previously learned set of color classes including ones for *field green*, ones for the dresses of the teams, and others, the visual perception module finds the regions of interest by first mapping the image pixels into the respective color class and then grouping the pixels that belong to the same color class into regions using morphological operations for noise elimination. Figure 3 shows the mapping of an image into the color class field green. The white pixels belong to this class the black ones not.

In order to find meaningful regions or blobs, in particular those that correspond to the objects of interest, the visual perception module applies morphological operations to the images resulting from color classifications.This step eliminates noise and makes regions more homogeneous.

In addition, we can characterize the objects of interest through properties of the image blobs they generate. In particular, using the assumptions that players are in upright positions and having contact with the field plane (at least, if we consider image sequences of particular length) we obtain very accurate estimates of the blob sizes based on the $\langle x, y \rangle$-coordinates of blobs in the image. Furthermore, the objects of interest, the players and the ball, have to satisfy certain compactness (ratio between area and perimeter). We can apply these assumptions to filter the data, and to more reliably extract the relevant objects.

Having color regions and color blobs as building blocks we can define complex regions of interest to which we might apply certain image processing operations. For example, in order to look for field lines in a focussed way we consider the image region that is field green, in the area where we expect the field but disregard those regions that are occluded by a player or the referee. This region can be expressed by: *($\neg$ green $\cap$ field region) - Team1Regions - Team2Regions - RefereeRegion*.

Robustness of Color Classification. Lighting conditions change when the camera sweeps from one side to the other, when the clouds change, when it begins to rain, etc. For these reasons reliable color segmentation cannot be achieved by learning color classes in advance and then hold them fix during the game. Rather they have to be adapted on the fly, in particular the color class *field green*. Thus we adapt the color class *field green* in an expectation maximization manner. In one step we use the estimates of the camera parameters to identify those regions that must be *field green* given the field model and the camera parameters. The relevant region is: *FieldRegion - Team1Regions - Team2Regions - RefereeRegion - neighborhoods of fieldlines ...* processed by some morphological operators to eliminate holes in the regions. We then take the pixels in these regions to estimate the color class *field green*. This color model is then used for the estimation of the camera parameters. In praxis, the class model can be estimated at a much lower rate than the camera parameters.

3.2 Estimation of Camera Parameters

In order to estimate the 3D position of the players and the ball, we need estimates of the camera parameters, that is its position, the direction it is pointed to, and the zoom parameter. Given the camera parameters and an assumption about the plane in which pixels lie the 3D coordinates of each pixel can be determined. In this section we will investigate the estimation of the camera parameters for a given image sequence.

We will do so by first describing our model of football fields. Then we look at the three-step iterative estimation process, detailed in section 3.2.2.

3.2.1 The Model of the Football Field

The purpose of the field model is the estimation of the exact position of the camera, of the direction it is pointed to, and the zoom of

the camera. Based on these camera parameters and the knowledge of the exact position and orientation of the field plane the visual perception module can accurately determine the $\langle x, y \rangle$ positions of the players on the field (see section 3.3).

The model of the football field is a set of 3D curves where each curve can either be a line or an edge. Edges are curves that separate two neighboring surfaces. Lines are combinations of two parallel curves where the area between the curves is visually distinct from the areas left and right of the line. The field lines are perfect examples of this line notion.

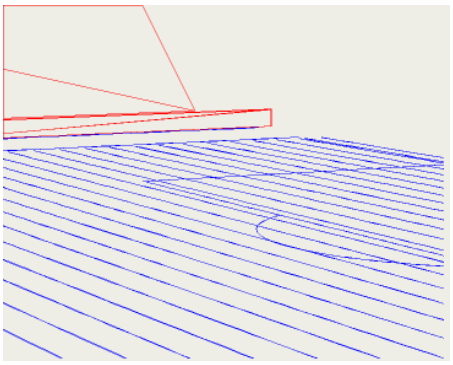

Figure 4: Field model used for estimating the camera parameters. Curve segments are depicted in blue and triangles in red.

The FIPM system uses the field model depicted in figure 4. The model contains the lines on the field. In addition, it contains the boards surrounding the football field and the stands. Both the boards and the stands are modeled as triangles with given position and orientation in space. Using these triangles the visual perception system can accurately determine the 3D position of each pixel that lies in the projection of such a triangle onto the image.

The boards and the stands are included in the field model because for many camera views the field lines alone do not suffice to unambiguously estimate the camera parameters. Using the boards and stands submodels the visual perception module can on the fly determine visual landmarks on the boards and in the stands and use them for estimating the camera parameters even if no field lines are in the camera view.

3.2.2 *Estimation Process*

We formulate the estimation of the camera parameters as an iterative optimization problem. The opimization process performs in each iteration three steps. In the first step the model is projected onto the image using the predicted parameter values. In the second step the optimizer searches for the image points that correspond to given model points. In the third step, the predicted parameter settings are modified such that we obtain the best match between the model and the image data. The search window for finding the best match is determined based on the expected innaccuracy of the predicted parameters. These three steps are depicted in figure 5.

Since the frame rate of the camera is about 25Hz (50 Hz interlaced) the change from one image to the following one is typically very small, the exceptions being camera sweeps following very long and hard passes. We will return to this issue later in section 3.2.3. So in the typical case, where the changes are small the predictions are also very accurate allowing for the use of small search windows.

The search methods for correspondences are specific to the different types of curves which allows for the realization of a high performance hybrid correspondence search. This means that we have curve segment specific "curve line" detectors. For example, the lines on the field are detected by looking for edges in the *field green* region, the edges resulting from lawn mowing look for points along the search perpendicular such that brightness variation left and right of the point becomes mininmal, that is the points to the left have a very similar brightness and so do the pints on the right. Other mechanisms for finding correspondences are taken for the edges between the field and the surrounding boards.

Given the correspondences we perform an optimization of the camera parameters such that the distance between corresponding points becomes minimal. This is done by a Newton iteration step. The system also estimates the covariance of the parameter values which captures the expected accuracy of the estimation outcome. The camera parameter values resulting from this optimization step is returned as the estimate of the parameters.

3.2.3 *Increasing Robustness*

In order to estimate the camera parameters reliably and accurately even if the field lines in the camera view do not suffice for an unambiguous estimation we generate visually distinctive landmarks on the fly and track them over sequences of camera images.

For this purpose, we first generate regions of interest by projecting the triangles of the field model onto the image. These are the regions where we can determine the 3D coordinates of the surface that generated the pixel. To these regions of interest we apply the feature detection and tracking method of Shi and Tomasi [14].

Even with these extensions the estimation of the camera parameters will sometimes fail, in particular when the camera man makes fast sweeps. To deal with failures in parameter estimation we run a monitoring process in parallel with the estimation process. This monitoring process signals a failure whenever the quality of the match between the predicted parameters and the parameters that result in the best local fit fall below a specified threshold. In this case a reinitialization of the camera parameters is triggered.

3.2.4 *Empirical Results*

At the current state of implementation the system can keep track of the camera orientation and the zoom parameter for up to 2,5 minutes, depending on the speed of camera motion and zooming and the visible components in the camera view. As far as we can tell the cases where the system looses track are caused by our model of the field beeing too inaccurate. In general, the length and the width of the field can vary substantially from stadium to stadium. An accurate and reliable estimate of these parameters from an image is not possible because if the camera view covers enough of the field then the field lines of the opposite side are blurred too much. Thus, so far we work with field models that are locally consistent with respect to the assumed camera position but not globally accurate. We currently, work on mechanisms that are less sensitive to the lack of globally accurate field models.

When successfully tracking the camera position the average accuracy of the field model being back projected onto the image is typically within a pixel occasionally 3-4 pixels. The system can automatically recover from lost tracks if the estimate is still within the convergence area of the model fitting procedure. For the remaining situations we currently develop a monitoring mechansm that detects lost tracks and reinitializes the system as soon as the camera view allows for an unambiguous match of the field lines.

At the moment processing an image can still take up to more than a second. We expect that by applying faster methods, and exploiting more prior information, we can reduce the processing time and bring the system close to frame rate.

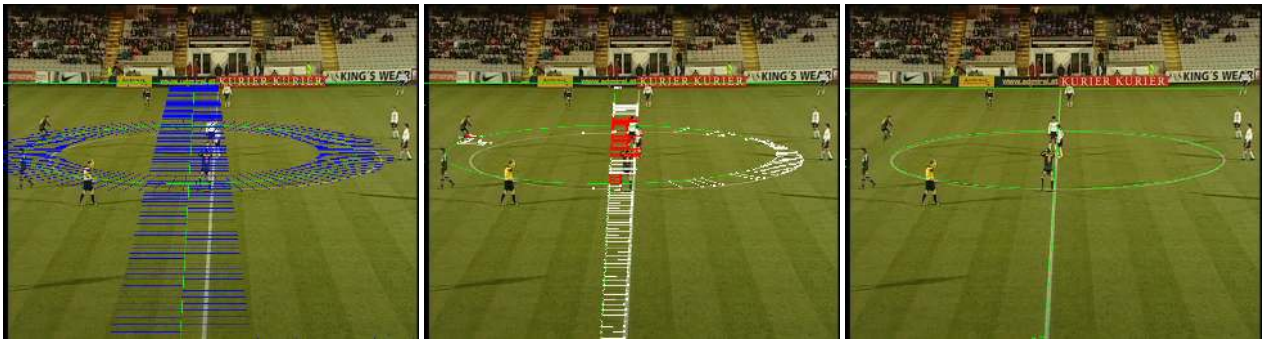

Figure 5: Estimating the camera parameters by taking the current parameters, projecting the field model using the current parameters onto the image (left), finding the correspondences between model points and the imagepoints (middle), and adjusting the camera parameters to achieve minimal errors (right).

3.3 Blob Recognition and Localization

The visual perception module gets all blobs in the field green region as its input. These are obtained through the regional expression: InRegion(field) $\cap$ $\neg$FieldGreen $\cap$ $min \leq Compactness \leq max \cap min \leq Size(blob, distance) \leq max$

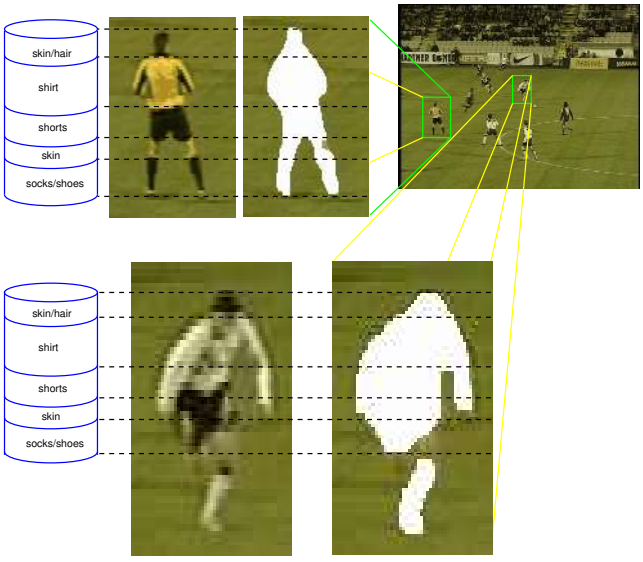

Figure 6: Player blobs segmentation. The player blob detection algorithm finds compact regions that are not field-green. The segmentation of players close to the camera is fairly accurate, but the situation is harder for the distant players because of the course grainedness, which causes some of their pixels to look field-green. Then they are segmented into two regions.

The next step is the blob classification. Blobs are classified into: the ball, the players of the one and the players of the other team and the referees. All other blobs on the field are ignored. The classification of the blobs into the different categories is straight forward. We first apply size constraints to the blobs and then identify the dress colors of the players. These color models are learned and updated during the game. The biggest complication are the partly occluded players that result in bigger blobs with possibly not identifiable color classes. These complications are resolved using the information provided by the object tracking system (see next section).

Another module monitors the blob sizes in the camera image. If the blob size is big enough, the blob identifier attempts to read the player number — if possible. If the player number is recognized the information is passed to the object tracker as evidence about the players' identity.

Given a blob that is inferred to be a player or ball the FIPM system estimates the $\langle x, y \rangle$ of the corresponding object on the field. To do so, we estimate the 3D position of the center of gravity of a player which can be much more reliably and accurately estimated than the position where the player stands on the field. The gain in accuracy and reliability has been empirically studied in several experiments.

Surprisingly, the player blob detection and localization was substantially less robust and accurate than we initially expected. This is explained by a number of reasons including the motions of the legs and the typical colors of skin and player dresses varying substantially. In particular as the players in the image are smaller and more distant a pixel in the image covers a much larger area in the world. Therefore many of the pixels depicting the player have a high fraction of field green and therefor cannot be segmented easily from the background.

To deal with these issues we have then implemented blob recognition using predictions made on the basis of previous observations (see next section). In this method the perception module receives hypotheses about players from the state estimation system and then validates and rejects these hypotheses based on the image. This way we can concentrate the player recognition mechanisms on selected regions of interest and use much more informative expectations about the blob size and the color distribution within the blob. Using this prior knowledge the accuracy and reliability of player recognition is increased substantially while at the same time reducing the computational cost for the player recognition.

4. PLAYER TRACKING

The detection of players and their classification based on single images is unreliable. Players might violate size restrictions when bending down or when colors get blurred due to fast camera motion. Sometimes players are hallucinated due to specular reflections in the field that make regions appear to be not field green.

4.1 Multi-Object Tracking

To reliably estimate the positions and motions of the players we apply probabilistic multi object tracking algorithms to keep track of player positions.

We use an extension of Reid's Multiple Hypothesis Tracking

(MHT) algorithm [11, 4] that has been further developed by Schmitt and his colleagues [12, 13]. Using probabilistic motion and camera models the MHT maintains probabilistic estimates of the players positions and update these estimates with each new observation. The computational structure of the algorithm is shown in Fig. 7. An iteration begins with the set of hypotheses of player states $H^k = \{h_1^k, \ldots, h_m^k\}$ from the previous iteration k. Each h_i^k is a random variable ranging over the possible positions of a player on the field and represents a different assignment of measurements to players, which was performed in the past. The algorithm maintains a Kalman filter for each hypothesis.

```
algorithm MULTIPLEHYPOTHESISTRACKING()
 1  let Ĥ^k = {ĥ_1^k, ..., ĥ_{m_k}^k}  % pred.hypos.
 2    Z(k) = {z_1(k), ..., z_{n_k}(k)}   % ob.feat.
 3    H^k = {h_1^k, ..., h_{o_k}^k }  % new hypos.
 4    X^{k-N}  % world state at time k-N.
 5  do for k ← 1 to ∞
 6    do Z(k) ← INTERPRETSENSORDATA();
 7       Ĥ^k ← APPLYMOTIONMODEL(H^{k-1}, M);
 8       for i ← 1 to n_k
 9       do for j ← 1 to m_k
10            do h_ij^k ← ASSOCIATE(ĥ_j^k, z_i(k));
11               COMPUTE(P(h_ij^k|Z(k)))
12       for j ← 1 to n_k
13       do H^k ← H^k ∪ {GENERNEWHYP(z_j(k))};
14       PRUNEHYPOTHESIS(H^k);
15       X^{k-N} ← {x_1^{k-N}, ..., x_{o_{k-N}}^{k-N}}
```

Figure 7: The multiple hypothesis tracking algorithm.

With the arrival of new sensor data (6), $Z(k+1) = \{z_1(k+1), \ldots, z_{n_{k+1}}(k+1)\}$, the motion model (7) is applied to each hypothesis and intermediate hypotheses $\hat{h}_i^{k+1}$ are predicted. Assignments of measurements to players (10) are accomplished on the basis of a statistical distance measurement, such as the Mahalanobis distance. Each subsequent child hypothesis represents one possible interpretation of the set of observed players and, together with its parent hypothesis, represents one possible interpretation of all past observations. With every iteration of the MHT probabilities (11) describing the validity of an hypothesis are calculated. Furthermore for every observed player a new hypothesis with associated probability is created (13).

Obviously, the heart of the MHT algorithm is the computation of the likelihood of the different hypothesis-observation associations, $P(h_{ij}^{k+1}|Z(k))$, in line 12 of the algorithm in Algorithm 7. Let Z^k be the sequence of all measurements up to time k. A new hypothesis of a player at time k is made up of the current set of assignments (also called an event), $\theta(k)$, and a previous state of this hypothesis, h_j^{k-1}, based on observed features up to time step $k-1$ inclusively. We can transform the probability of a player's hypothesis $P(h_i^t|Z^k)$ using Bayes' rule and the Markov assumption in an expression that can be obtained more easily:

$$
\begin{aligned}
P(h_i^k|Z^k) &= P(\theta(k), h_j^{k-1}|Z(k), Z^{k-1}) && (1)\\
&= P(\theta(k), h_j^{k-1}|Z(k), H^k) && (2)\\
&= \alpha * P(Z(k)|\theta(k), h_j^{k-1}, Z^{k-1}) && (3)\\
&\quad P(\theta(k)|h_j^{k-1}, Z^{k-1}) P(h_j^{k-1}|Z^{k-1})
\end{aligned}
$$

According to Bar-Shalom and Fordham [1], this equation can be transformed into an expression over a set of parameters, and they allow us to fine tune the calculation of probabilities to exploit the domain constraints. These parameters include ones for prior probability mass functions of the number of spurious measurements and of new detection of players, probabilities of detection and termination of a track originating from a particular hypothesis, model the declination of an unobserved hypothesis probability over time, and total numbers of spurious measurements and new detections of players. The exact form of the transformed data association probability and a more thorough explanation of its terms and parameters can be found in the papers by Cox and Hingorani [4] and Schmitt et al. [13].

In our tracking system we dynamically modify these parameters for each track and measurement according to the relevant situation. For example, it is highly unlikely that a player disappears while being well observed by the camera and in the middle of the field. The likely explanation is that the player is being obstructed by another one and the detection module is having problems detecting him/her. In this case, we set the parameters accordingly to favour the continuation of the track without measurements, hoping to detect the player in the next images. See section 4.2 for further discussion of the used constraints.

4.2 Exploiting domain constraints

The nature of our tracking task allows us to make assumptions that do not hold in the general case of multi object tracking. Under these assumptions we are able to substantially increase the robustness, accuracy, as well as reduce the required computational resources by specializing the estimation and inference techniques to the characteristics of the application. We will list some of these assumption that result in substantial improvements of the algorithm below.

The first and obvious way of increasing the performance of the MHT is to set the parameter of the formula in situation specific ways. For example, players cannot magically appear or disappear in the middle of the camera frame. This constraint is implemented by adjusting the average time of new track appearance and track termination. New tracks are to be primarily created at the boarders of the camera frame. The same holds for the termination of tracks. Making these assumptions enables the system to substantially prune the hypotheses tree and therefore allows for more thorough analysis of the individual hypotheses. This is done by maintaining hypotheses trees of greater depth.

Another improvement is the handling of occlusions of one player by another one. This could be handled by increasing the time that a track survives without being supported by additional observation. A more controlled way of "remembering" occluded players is to generate virtual observations from hypotheses that the system knows to be occluded. A similar mechanism was used by Fox and his colleagues [6] to avoid collisions with known obstacles that were invisible for the robot sensors.

Another twist of the tracking problem is that we can assume that the players all move in the same 3D plane. Under these conditions a player is closer to the camera if and only if his position in the image is lower than that of the other one. This qualitative relationship holds independently of the inaccuracy of our estimate of the camera position and can be exploited to reject many data associations that are impossible with respect to this constraint.

These and other modifications to the MHT algorithm allow for reliable and accurate player tracking in situations where the more general version of the algorithm is doomed to fail [7].

At the moment our algorithm generates maximally extended trajectories for which the system is sure that they are generated by the same player. The player identity itself is specified manually.

In particular the semi-automatic inference of the player identity will be a focus of our future research. The tracking system will use prior distributions of player positions, knowledge about the player roles, occasionally read player numbers, the identity of players that are known in order to estimate a probability distribution over the identity of a given but unknown player.

5. MOTION AND ACTION RECOGNITION

Motion and action recognition abstracts away from the details of the estimated position data and the track segments generated by the Multi Hypothesis tracking algorithm. The raw data are transformed into compact motion models and then classified into ball actions [2]. These computational tasks are detailed in this section.

5.1 Motion Interpretation

The motion interpreter computes motion models for the players and the ball o that are sequences of motion segments $m = m_1, m_2, ..., m_n$. The trajectory of an object is represented as a piecewise defined function f that maps time indices into the respective *x* and *y* coordinates of the object. The function f is defined for all t with $t_1 \leq t_2$ 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 number), t_1, t_2 the starting and end time point of the motion segment, and p_1, p_2 the respective start- and end position.

The motion model also contains the set of all ball contacts, ball out of bounds, and referee whistles (game interruptions and continuations) as instantaneous events. These events are asserted manually. Ball contact events are those where the ball is accelerated due to the action of a player. Ball out of bounds events occur if the ball leaves the football field.

As the motion model we have chosen piecewise linear models. This model is very efficient to compute and the motion tuples are very compact. The linear modeling of motions also simplifies the computation of distances, intersections, and other derived information. The accuracy can, as in the other models be adjusted through the segmentation thresholds.

The motion interpreter maps the position data into a sequence of motion tuples, where each motion tuple describes a uniform motion segment. The interpreter works iteratively. If it has a partial segment m_i it represents this partial segment as a motion function. It then uses this motion function to predict the next position p_j. If the difference between the predicted and the observed next position is smaller than a threshold ϵ_1 then m_i is extended to cover p_j and the motion function is updated accordingly. Otherwise m_i is asserted to the motion model as a complete motion segment and a new motion segment m_{i+1} is started.

Instantaneous motion events are currently recognized mainly manually. Referee whistles are explicitly asserted. Out of bounds events are detected by intersecting the linear motion segments with the field lines. Finally, the ball contacts are asserted by hand.

5.2 The Episode Model

The motion model is further abstracted into the episode model.

In the episode model we consider the episodes of ball movement to be primary. This is natural because people watching football and summarize plays in football games primarily as a sequence of ball actions. Actions away from the ball are included only if they become relevant for the ball actions later.

We distinguish three different classes of ball actions: keeping control over the ball, passing, and shooting. 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.

Unfortunately, the high technical difficulty of playing football makes such simple classification rules very inaccurate. Control over the ball, in particular with opponents around, is very difficult and therefore actions do not only have nondeterministic effects but often fail completely: the player looses possession of the ball. It is often impossible to distinguish whether a lost ball is caused by a bad pass, shot, or dribbling. In our approach we deal with these ambiguities with probabilistic reasoning techniques.

We consider an *episode* e to be a triple $\langle \langle m_i, ..., m_j \rangle, se, fe \rangle$ where $m_i, ..., m_j$ is a sequence of motion segments, se is the starting and *fe* the finishing event. To be an episode the triple must satisfy the following requirements: (1) m_i is started by ball contact se of player p. (2) m_j is ended by *fe*, which is of the type ball contact, referee whistle, or out of bounds. (3) All events e that occur are ball contacts of player p. (4) $m_i, ..., m_j$ is the maximum sequence of segments that satisfies (1) to (3).

For the recognition of episodes we use a small finite automaton that starts a motion segment sequence if it receives a motion segment starting with a ball contact of player p. The automaton stays in its intermediate state as long as the motion events are ball contacts of player p. It terminates for all other motion events. All accepted motion sequences are episode candidates.

The episode model differentiates between passes, shots, and extended ball possessions of a single player. The main problem is the classification of football actions that have failed. To deal with ambiguities in the classification of failed actions we apply classification rules that assign classes to episodes with a subjective probability. To get a principled way of stating such rules we apply decision tree learning . Given a set of classified examples a decision tree learning algorithm learns a set of classification rules with the objective of making the rules as general and accurate as possible. Decision tree learning algorithms also estimate the expected accuracy of the classification rules.

When we apply decision tree learning to the acquisition of classification rules for football actions two problems must be solved. First, we need a suitable feature language for representing episodes such that their representation correlates well with their classification. Second, we must provide the examples that we need for learning. The main problem here is that we need to know the intentions of the players to provide examples of ball actions and their classification. For the feature language we have used the following features: the duration, the number of ball contacts within the episodes, whether ball possession changed from one player to another one, from one team to the other one, the dominant motion direction, and the average velocity of the ball.

We obtained the best results by classifying at the toplevel into the action classes pass, dribbling, and shot and adding one additional class "lost ball" that comprises the episodes where ball possession goes from a player to the other team, from a player to the out, and episodes interrupted by referee whistles. In this approach, we have the advantage that we can specify all toplevel classes crisply. The top-level classification rules are complete and classify uniquely. That implies that every episode candidate is classified as an instance of exactly one top-level class. The biggest assumption is that the classification assumes that if a football action was successful the player has intended it. We believe that by making this assumption the resulting accuracy is higher than it would be when learning the classes from examples. We will reevaluate this assumption when we have comprehensive data from real games.

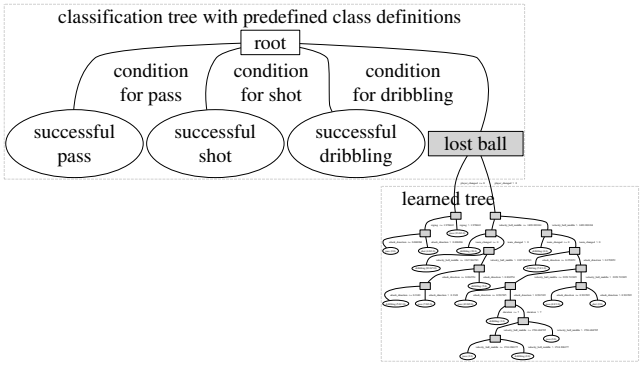

Figure 8: Classification tree. To classify the ball actions into different action classes we use a classification tree consisting of defined conditional leafs for successful actions and a learned decision subtree for lost balls.

In this approach we apply decision tree learning only to the class of lost ball possession. A sample classification rule that we obtained is: A failed pass is an episode in which the ball is played to the front but not in the accurate direction of the opponent goal, the ball velocity is smaller than that of shots, and the ball possession changes from one team to the other.

6. RELATED WORK

A number of observation systems for football games and sport games have been developed, most of them in the sport sciences and some of them even commercially. Those systems are typically characterized by their requirements for extensive manual data entry by operators. Intille and Bobick [9, 8] have developed a seminal visual observation system for American Football and this is the closest approach to ours. Their system include mechanisms for action, formation, and play recognition. Differences are caused by the different natures of the game. American football is structured into modular, preplanned plays with failed actions (interceptions and turnovers) being exceptions, players having very specific roles in plays, and the ball being held most of the time. In the real football these characteristics are not met and complicate the visual observation drastically. Another difference is our emphasis on accurate estimation of player positions which is the key for the recognition of game situations such as scoring opportunities, players being under pressure, passing opportunities, etc.

The research work in computer vision that is applicable to game observation is too big to be discussed in detail. Perhaps most relevant is the work of Malik's and Forsyth's vision group. In particular ideas of their work on recognizing actions at a distance [5] and the learning and application of appearance models for tracking human action [10] have been incorporated into our system.

7. CONCLUSIONS

This paper has described the nuts and bolts of observing multi agent systems for analyzing agent behavior. We have considered a particular application domain: the visual observation of football games. Football, in particular in the disguise of RoboCup has become a challenging testbed for agent technology.

The observation and interpretation system described in this paper is a part of a larger system. The analysis of Football games based on position data is discussed by Beetz, Kirchlechner and Lames [3]. Kinds of action models that are proposed in this paper are also used for making autonomous robots "action-aware" [15].

We plan to showcase the vision-based game observation system for games of the Football World Championship 2006 in Germany.[1]

8. REFERENCES

[1] Y. Bar-Shalom and T. Fortmann. Tracking and data association. Academic Press., 1988.

[2] M. Beetz, S. Flossmann, and T. Stammeier. Motion and episode models for (simulated) football games: Acquisition, representation, and use. In *3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS)*, 2004.

[3] M. Beetz, B. Kirchlechner, and M. Lames. Computerized real-time analysis of football games. *IEEE Pervasive Computing*, 4(3):33–39, 2005.

[4] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(2):138–150, 1996.

[5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[6] D. Fox, W. Burgard, S. Thrun, and A. Cremers. A hybrid collision avoidance method for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[7] I. Horswill. Analysis of adaptation and environment. *Artificial Intelligence*, 73(1-2):1–30, 1995.

[8] S. Intille. *Visual Recognition of Multi-Agent Action*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

[9] S. Intille and A. Bobick. Recognizing planned, multi-person action. *Computer Vision and Image Understanding*, 81:414–445, 2001.

[10] D. Ramanan and D. Forsyth. Finding and tracking people from the bottom up. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, 2003.

[11] D. Reid. An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control, 24(6):843–854, 1979.

[12] T. Schmitt, M. Beetz, R. Hanek, and S. Buck. Watch their moves: Applying probabilistic multiple object tracking to autonomous robot soccer. In *The Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[13] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18(5), October 2002.

[14] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[15] F. Stulp and M. Beetz. Action awareness – enabling agents to optimize, transform, and coordinate plans. In *Accepted for the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.

[1]The research reported in this paper is partly funded by the Deutsche Forschungs Gemeinschaft.