An Adaptive Vision System for Tracking Soccer Players from Variable Camera Settings

Suat Gedikli, Jan Bandouch, Nico v. Hoyningen-Huene, Bernhard Kirchlechner, and Michael Beetz

Intelligent Autonomous Systems Group,
Technische Universität München, Munich, Germany
aspogamo@cs.tum.edu
http://aspogamo.cs.tum.edu

Abstract. In this paper we present ASPOGAMO, a vision system capable of estimating motion trajectories of soccer players taped on video. The system performs well in a multitude of application scenarios because of its adaptivity to various camera setups, such as single or multiple camera settings, static or dynamic ones. Furthermore, ASPOGAMO can directly process image streams taken from TV broadcast, and extract all valuable information despite scene interruptions and cuts between different cameras. The system achieves a high level of robustness through the use of modelbased vision algorithms for camera estimation and player recognition and a probabilistic multi-player tracking framework capable of dealing with occlusion situations typical in team-sports. The continuous interplay between these submodules is adding to both the reliability and the efficiency of the overall system.

1 Introduction

The automatic observation of team-sport games is gaining increased attention for different reasons. As such games, particularly soccer, are enjoyed by millions, television broadcasting companies are constantly seeking new ways to improve the viewers' experience. During the last FIFA World Cup in Germany, highlights and critical scenes were virtually reenacted after the game, accompanied by expert discussions and analyses. Animations and replays of goals viewable from arbitrary virtual camera positions were made available on the internet some hours after the game. Such recreations of game scenes are done mostly manually, and require a massive amount of human intervention for estimating the needed 3D position data of the players and the ball. An automatic acquisition of this data could drastically speed up the recreation while cutting down costs. The position data could also be used to save bandwidth when broadcasting soccer games to mobile devices, where scenes could be redisplayed using a game-like computer graphics API.

Trainers and major sports clubs are interested in statistics and analyses of their own as well as the opposing team. Having a detailed representation of a game facilitates the automated study of tactics and weak spots. A number of top teams already equipped their stadiums with semi-automatic player tracking systems. However, the downside of these commercially available systems to date is the need for an expensive hardware setup bound to a single stadium, as they rely on multiple static cameras that need an exact calibration of their positions and parameters.

Scientifically, abstract and symbolic representations allow to analyze games and human activities in a manner not able until today. This holds for sports as well as computer scientists. For instance, the huge amount of position data representing real-world multi-agent activities without artificial patterns usually found in simulated data provides interesting opportunities for AI research. The acquisition of such 3D position data poses interesting problems in the computer vision domain, amongst them initialization and tracking of camera models, the reliable realtime recognition of small-sized players, the tracking of multiple players through occlusions and automated scene analysis.

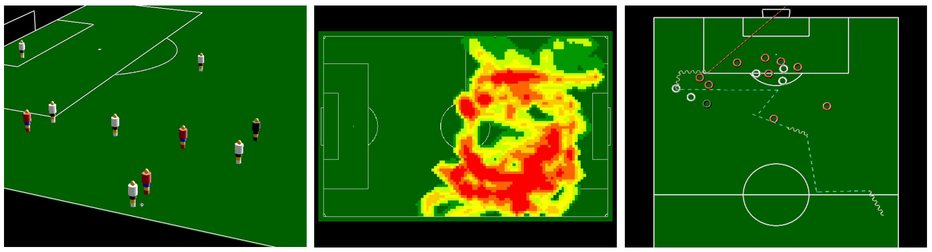

Fig. 1. Example applications made possible using ASPOGAMO: virtual camera views (left), statistical heatmap showing main player distributions (center), tactical scene analysis (right)

In this paper we present ASPOGAMO (Automated SPOrt Game Analysis MOdel), a computer vision system that is able to extract consistent motion trajectories of soccer players from camera recordings automatically and reliably with little human interaction. One of the main contributions of ASPOGAMO to date is the integration of several state-of-the-art computer vision algorithms to form a robust and reliable system that can deal with low-quality input, bad lighting situations and game-typical occlusions while still running almost in realtime on standard desktop PCs. Another contribution is its adaptivity with respect to the source of input: (1) it can run using only the standard TV broadcast stream, gathering the required data despite cuts between different cameras or scene interruptions like close-ups or replays; (2) it can deal with uncalibrated cameras constantly changing zoom and direction while pointing towards the game action; (3) if wanted, multiple synchronized cameras can be used to improve both the field coverage and the tracking accuracy through sensor fusion. This adaptivity makes ASPOGAMO flexible and easy to set up without special requirements. Figure 1 shows some example applications that were developed using the motion data generated by ASPOGAMO.

Due to the space limits in this paper we will try to provide a comprehensive overview of ASPOGAMO without giving in-depth explanations of the underlying algorithms. We start in Chap. 2 with an introduction to the system architecture, and then describe in more detail the *Vision Modules* (Chap. 3). Chapter 4 addresses the creation of consistent player trajectories. We finish with experimental results (Chap. 5), related work (Chap. 6) and our conclusions (Chap. 7).

2 System Architecture

The general architecture of ASPOGAMO is shown in Fig. 2. All modules are running as standalone processes and communicate through well defined interfaces. Being designed on an abstract level, modules can easily be exchanged with other implementations. CORBA is used as intermediate layer for communication which enables the system to be flexible, distributed and scalable.

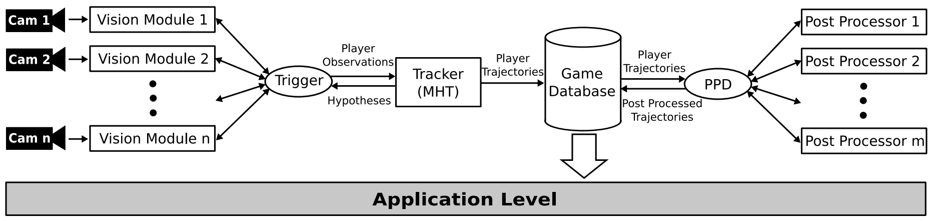

Fig. 2. Architecture of the ASPOGAMO system

The system is designed to process arbitrary camera input streams consisting of recorded footage from either static or dynamic cameras. Furthermore, input streams recorded from TV broadcast can be used. Each stream is associated with a *Vision Module*, whose main task is to determine possible player positions and their uncertainties in the world coordinate frame and forward these observations to the *Tracker*. As the *Vision Modules* are running asynchronous (probably distributed on different machines), and the *Tracker* expects the observations in the right temporal order, an intermediate *Trigger* module is used to abstract and hide the acquisition of player observations from the trajectory generation. It synchronizes all observations received from registered *Vision Modules*, forwards them to the *Tracker*, and sends back the predicted object hypotheses for the next timestep. The *Tracker* generates consistent trajectories according to the observations and its internal predictions and writes them into the *Game Database*. The *Game Database* is the access point for all higher level applications depending on the data generated by ASPOGAMO. Because player trajectories generated by the *Tracker* might be split or incomplete, they can be post-processed via a GUI in the *Post Processor* modules. Apart from editing the trajectories directly, users can also add metadata like player identifications. Due to the distributed nature of our system, an arbitrary number of these modules can be started simultaneously, even when the trajectory generation is still running. The *Post Processing Distributor* (PPD) is responsible for keeping the *Game Database* and the *Post Processor* modules consistent all the time.

3 Camera-specific Vision Modules

The *Vision Modules* are responsible for generating player observations from the input video streams. An observation consists of a player position on the field plane including uncertainty and the team affiliation. Each module is responsible for processing all scenes recorded by a single *physical* camera associated with the module. To be able to calculate the player positions on the field plane, the intrinsic and extrinsic parameters of this camera need to be estimated. In the case of dynamic cameras, parameters like *pan*, *tilt* and *zoom factor* are constantly changing and have to be determined for each frame. Furthermore, when a broadcasted input stream with scene cuts is provided, scenes processable by the module have to be identified.

The internal design of the *Vision Modules* is shown in Fig. 3. A continuous scene from the input stream is processed frame by frame in the *Scene Processor* by first estimating the unknown camera parameters (Sect. 3.2) and then detecting the players (Sect. 3.3). If a shot cut is found in the input stream, scene processing is stopped and the *Scene Filter* selects the next processable scene.

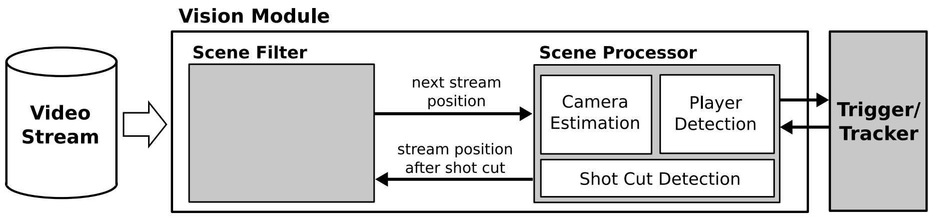

Fig. 3. Design of the Vision Module

3.1 Initialization and Shot Cut Handling

Before starting the image processing, a couple of initializations need to be performed. First, the constant intrinsic and extrinsic camera parameters need to be estimated, along with the field size which (according to FIFA regulations) may vary and usually is unknown. Furthermore, as the utilized vision algorithms heavily rely on colors (because colors are chosen to be visually distinctive in soccer games), color models aligned to the *physical* camera's color perception are precomputed to speed up processing.

Constant Camera Parameter Estimation: The estimation of all camera parameters is difficult, as there is usually no possibility to use standard calibration methods without access to the original cameras. The intrinsic parameters (except the principal point but including radial distortion), the position of the camera and the field size are thus estimated semi-automatically and simultaneously by selecting several corresponding field positions out of different views of the respective camera. Uncertainties are assigned to the correspondences by the user specifying an ellipse around the image-points defining a 95% quantile. A system of equations is set up and solved using least squares optimization considering the uncertainties. Besides the needed parameters, its covariances are also estimated. To get the best results, the selected views should correspond to all sides of the field.

Color Model Initialization: Color classes are modeled as mixture of Gaussian in RGB space, which allows to model e.g a field with a cast shadow from a stadium roof. For efficiency reasons, the color space is compressed to 16bit, so that lookup tables with Mahalanobis distances and densities can be precomputed. Color classes are learned semi-automatically using clustering methods, and their distributions are readapted constantly during the game to account for lighting changes.

Shot Cut Handling: After initialization, scene processing on the input stream is started. If the input stream is interrupted by shot cuts (Fig. 4), these are detected from inconsistencies in the optical flow generated during camera parameter estimation (Sect. 3.2). The processing is stopped, and the *Scene Filter* tries to find the next scene that can be processed. To do so, a given scene is classified as *slow motion*, *out of field view* or *field view*. By looking at the color histogram and the number of pixels classified as field color, *out of field views* and most *close-up views* can be skipped. *Slow motion* segments that are out of the timeline and thus of no value are detected using a zero-crossing measure [10]. The remaining candidate scenes are evaluated if they match with the given camera model. Currently this is done through user interaction, but we are working on an automatic matching process using segmented field lines. If a match is found and the dynamic camera parameters (pan, tilt, zoom) are reinitialized, processing is once again handed over to the *Scene Processor*.

In case of continuous input streams, no shot cuts will be detected so that the processing runs uninterrupted. If a broadcasted image stream consists of more than one evaluable camera view, one *Vision Module* for each camera must be registered and assigned the same input stream.

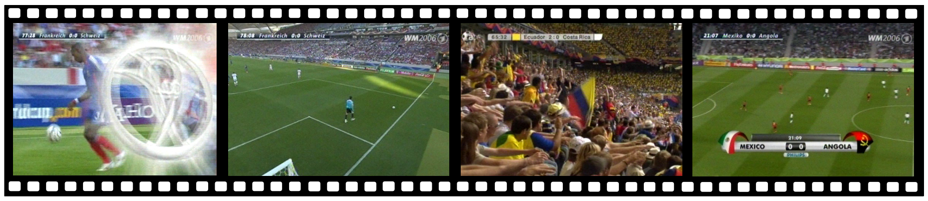

Fig. 4. Scenes contained within broadcasted image stream

3.2 Camera Estimation

As already mentioned, not all camera parameters stay constant during the game. Standard TV cameras track the game actively, and are continuously changing their pan and tilt orientations as well as the zoom factor (focus). Even the position of the optical center changes, when the camera is rotated on a tripod, but the parallax effects are negligible due to the big distance of the camera from the field. ASPOGAMO uses a modelbased localization for estimating the missing parameters in each frame. A virtual model of the field is projected into the image, and an objective function made up from the real-world distances between virtual field points and corresponding image points is minimized to obtain the best match. Because the estimation is done frame by frame and the camera motions are fluid, this can be described as a tracking problem and solved using an Iterative Extended Kalman Filter (IEKF).

Correspondences for the objective function are found by projecting points on virtual field lines into the image using predicted parameter values. The corresponding image points are searched in perpendicular directions to the virtual lines inside a search window that is determined according to the parameter uncertainty. Due to inhomogeneous lighting conditions, low resolutions, bad image quality and fast camera movements (Fig. 5), field lines (especially distant ones) are hard to identify using uninformed line detectors. Field lines do not appear as white, but as slightly lighter green. Using our probabilistic mixture of Gaussian color models, we reduce the search for line correspondences to the problem of finding the most probable transition between the field color class and the line color class within the search window. Matches are thresholded depending on the expected linewidth in pixels, determined with the predicted camera parameters. Covariance matrices of the correspondences are estimated heuristically by their probability and uniqueness.

To improve the prediction of the parameters from the IEKF, we additionally estimate the homography describing the perspective transformation between two subsequent images. This transformation maps points between two images regardless of their real 3D positions. It is estimated by calculating the optical flow with the *Lucas Kanade* method to find 100 point-matches around the image border. To eliminate outliers, we determine the homography using the RANSAC algorithm [3] and remove all non-confirming matches. All other matches are used to find the relative motion of the camera with the M-Estimator [12] using the weight-function from Huber. Predictions generated solely from the homographies are accurate enough to stay within the convergence-rate of the estimator for about 100-150 frames. Thus they can be used to temporarily track the camera parameters when not enough line correspondences are found, i.e. when no field lines are actually visible. Cases where the drift of the field model is unavoidable are detected by the *Shot Cut Detection*, and a reinitialization is triggered.

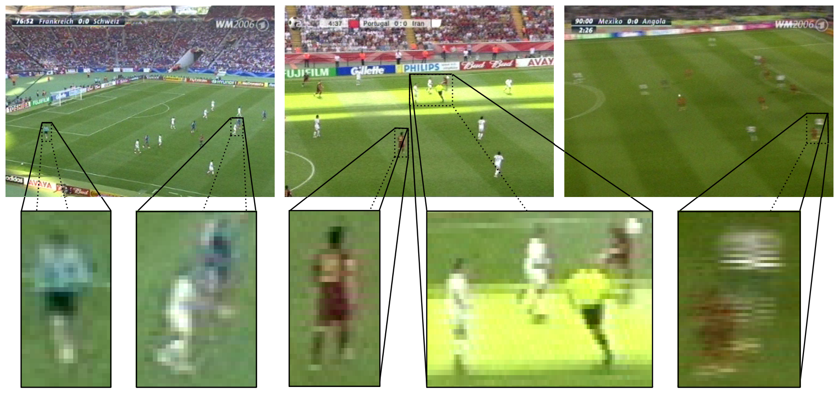

Fig. 5. Hard scenes that have to be dealt with: Small player sizes (left), field with cast shadow (center), camera motion blur (right)

3.3 Player Detection

As seen in Fig. 5, the detection of players is a difficult task. Typical player sizes in the image are below 30 pixels height, players often occlude each other, and shapes and colors tend to be blurred and nondistinctive. Furthermore, the detection should run in realtime, but images need to be processed at full resolution because of the small player sizes.

We use a number of probabilistic perceptual cues that can be calculated easily using knowledge of the domain. The processing steps are visualized in Fig. 6. First, all potential player regions that are lying inside the field are segmented based on our color model (Fig. 6a). Then, we use special templates to calculate likelihood-maps for player locations based on color distributions, compactness of the segmented region and the vertical spacing inside the regions (Fig. 6b). These likelihood-maps are fused, and predictions generated by the tracker are integrated (Fig. 6c). Player detections are then generated from the strong modes in the combined likelihood-map.

The segmentation of the player regions is achieved by first segmenting the field using the precomputed Mahalanobis distances from the field color class. Morphology is used to remove noise in the segmentation and to combine split field regions. The field region is then inverted and intersected with its convex hull, yielding the potential player regions that may consist of one or more persons, none or only part of a person. The templates used for the calculation of the likelihood-maps are scaled to the expected size of a player at a specific image position according to the predicted camera parameters. The template used for color template matching is divided into three square regions, one for the shirt color, one for the short color and one for the socks color of each team. The use of square regions allows to optimize the calculations using integral images. Furthermore, the shirt region can be shifted horizontally, so that inclined players are also detected. The template is moved across every pixel in the potential player regions, and a likelihood is calculated based on the Mahalanobis distances of the colors inside the template to the color classes that have been associated to the template. Another likelihood is calculated based on the number of non-field pixels inside the template. This ensures that the overlap between the player template and the segmented region is high, which we call the *Compactness Constraint*. The last likelihood calculated is called the *Height Constraint*, and produces high likelihoods near locations where either the distance to the upper or the lower contour of the segmented region is near the optimum for the expected player size. This improves the accuracy of the player localization. After fusing the calculated likelihoods with the hypotheses generated by the tracker, the detected observations are transformed into real-world coordinates on the field plane and sent to the tracker.

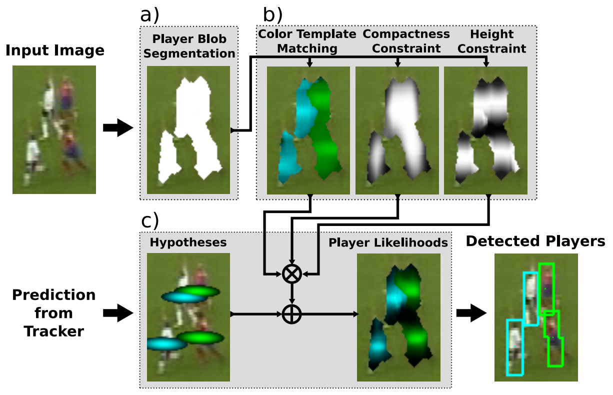

Fig. 6. Player Detection: a) rough segmentation of non-field blobs b) calculation of likelihood-maps based on color template matching, compactness and height constraints c) fusion of likelihood-maps and integration of tracker predictions

4 Trajectory Generation

The *Vision Modules* described in the previous section provide observations of players, i.e. positions on the field plane including uncertainties and the team affiliation. So far, these observations lack any temporal correlation. Furthermore, players can be missed, e.g. if temporarily occluded, and the observations might be erroneous or imprecise. It is therefore necessary to combine observations into consistent trajectories belonging to single players.

ASPOGAMO uses a Multiple Hypothesis Tracker (MHT) [9] to automatically create affiliations between observations and previous trajectories using the positions and covariance matrices of the observations. The MHT can keep different hypotheses during ambiguous situations (mainly resulting from occlusions) to resolve these situations when time progresses. The maximum time interval allowed for keeping hypotheses depends on the quality and distinctiveness of the observations (we are currently using a maximal depth of 15 frames). Observations made by multiple cameras can be used to improve accuracy through fusion of corresponding observations that is inherently supported by the MHT. A more detailed description of the player tracking can be found in Beetz et al. [1].

Once a trajectory is assured or can no longer be continued, i.e. a player has left the camera field of view or an ambiguous situation could not be resolved, the trajectory is written to the *Game Database*. For some applications it might be necessary to add additional information to a trajectory, e.g. the name and ID of a player. Such assignments can be done using the *Post Processor* modules. It is also possible to join, split or delete trajectories. We are currently researching ways to extract metadata such as player IDs automatically using a combination of visual and acoustical cues.

5 Experimental Results and Applications

ASPOGAMO has been under public display during FIFA World Cup 2006 in Germany. Extensive experiments have been conducted with input streams broadcasted live during the World Cup (more than 40 games) as well as other input streams retrieved either unaltered from TV cameras directly or from our own multiple camera setups.

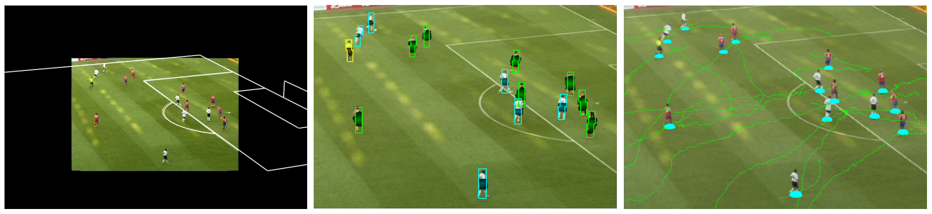

Fig. 7. Camera estimation (left), player observation (center) and trajectory generation (right)

Using original television camera streams without shot cuts, the camera estimation is able to track the camera parameters up to 20 min without the need for reinitialization. The mean time for tracking the main center camera without getting lost is about 8 min, while other cameras from broadcast having more abrupt motions could be tracked between 1 min and 12 min in our experiments. Using the broadcasted streams with shot cuts, several goal scenes from World Cup have been tracked without getting lost. Most of the position data from the broadcasted TV stream with shot cuts from the opening game between Germany and Costa Rica has been written to our *Game Database*. The detection rates for player observations are ranging above 90% for standard scenes, and still above 70% for especially crowed scenes like corner situations (Table 1). Most of the missed detections and misclassifications are resulting from occlusions, and are later resolved in the MHT. The trajectories for the generated scenes usually last as long as a player is in camera view, unless there are ambiguous occlusion situations in heavily crowded scenes that couldn't be resolved by the MHT. In such a case, trajectories are split and have to be joined manually during post-processing. In one experiment we recorded a soccer game with a HDV (high definition) camcorder that could cover most of the field, and got single player trajectories lasting for more than 5 minutes, before the spectators in front of the camera jumped up and spoiled the scene. In other experiments we were successfully tracking games with difficult lighting situations such as cast shadows on the field (e.g. Portugal vs. Iran).

game	player count	missed players	misclassified	false positives
Germany - Costa Rica	2353	4.78%	0.38%	2.37%
Argentina - Serbia-Montenegro	1605	5.51%	1.88%	1.23%
Portugal - Iran	592	21.96%	3.72%	0.34%
Corner Situation	462	17.53%	3.68%	6.06%

Table 1. Manually verified detection rates for player observations

To test the constant camera parameter estimation, we have recorded a soccer match with 4 calibrated cameras and measured the camera positions and the field size using laser as ground truth. Table 2 shows the resulting mean errors in the estimation.

	pos-x	pos-y	pos-z	field-width	field-length
field size known	0.19 m	0.41 m	0.43 m	-	-
field size unknown	0.19 m	1.68 m	0.34 m	0.68 m	1.87 m

Table 2. Mean estimation errors of position and field size estimation

The inaccuracies of the camera estimation range from about 5 cm to 1.0 m, determined through back-projections on the field-model. The inaccuracies of the player detection are typically lower than 0.5 m, and have been estimated by comparison with manually determined player positions on the field-model. Position inaccuracies increase with the distance of the players from the camera. When fusing observations from multiple cameras, position accuracy is improved, as the uncertainties decrease. Similar effects are achieved when smoothing the trajectories in post-processing. Figure 7 shows visualizations of the camera estimation, the player detection and the final output tracks.

We have created some videos showing example applications that can be generated from the position data. Figure 1 shows some screenshots. The full videos can be downloaded from the ASPOGAMO webpage at `http://aspogamo.cs.tum.edu`.

6 Related Work

Some relevant work in sports video processing is surveyed by Yu and Farin [11]. Other recent contributions are Kang et al. [6], Figueroa et al. [2] and Nillius et al. [7], but they are bound to static cameras. This drastically simplifies both the camera estimation and the player recognition. Hayet et al. [4] and Okuma et al. [8] (tracking hockey players) use dynamic camera settings, but lack discussions of the reliability, efficiency and accuracy of the tracking. Related work can also be found in the domain of visual surveillance (see Hu et al. [5] for a survey), although different assumptions are made in team-sport games than in surveillance scenarios (known team/field colors).

7 Conclusions

We have presented ASPOGAMO, a vision system used for retrieving the motion trajectories of all visible players during a soccer game. ASPOGAMO reliably handles a number of difficult situations like fast moving and zooming cameras, occlusions and changing lighting conditions while still being efficient and requiring only little human interaction. Being capable of processing inputs from different camera settings, ASPOGAMO meets many demands of a variety of applications. It is possible to create a complete and accurate motion trajectory database of a 90 minutes game fully covered by multiple cameras, or to extract all informative position data from a broadcasted stream on TV with shot cuts. The retrieved motion data can be used for tactical and statistical analyses, for inferring abstract activity models of players and teams, and for enhancing broadcasted scenes with virtual information of scene replays.

References

1. M. Beetz, J. Bandouch, S. Gedikli, N. von Hoyningen-Huene, B. Kirchlechner, and A. Maldonado. Camera-based observation of football games for analyzing multi-agent activities. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.
2. P. Figueroa, N. Leite, R. Barros, I. Cohen, and G. Medioni. Tracking soccer players using the graph representation. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, pages 787–790, 2004.
3. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
4. J. Hayet, T. Mathes, J. Czyz, J. Piater, J. Verly, and B. Macq. A modular multi-camera framework for team sports tracking. In *Proc. of IEEE Conf. on Advanced Video and Signal-Based Surveillance (AVSS'05)*, pages 493–498, 2005.
5. W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, 34(3):334–352, August 2004.
6. J. Kang, I. Cohen, and G. Medioni. Soccer player tracking across uncalibrated camera streams. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (PETS 03), 10 2003.
7. P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking - linking identities using bayesian network inference. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2187–2194, Washington, DC, USA, 2006. IEEE Computer Society.
8. K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV (1)*, pages 28–39, 2004.
9. D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.
10. Y.-Q. Yang, W. Gu, and Y.-D. Lu. An improved slow motion detection approach for soccer video. In *Proceedings International Conference on Machine Learning and Cybernetics*, volume 8, pages 4593–4598, 2005.
11. X. Yu and D. Farin. Current and emerging topics in sports video processing. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.
12. Z. Zhang. Parameter estimation techniques: a tutorial with respect to conic fitting. *Image and Vision Computing*, 15(1):59–76, January 1997.