Broadcast Court-Net Sports Video Analysis Using Fast 3-D Camera Modeling

Jungong Han, Dirk Farin, *Member, IEEE*, and Peter H. N. de With, *Fellow, IEEE*

***Abstract*—This paper addresses the automatic analysis of court-net sports video content. We extract information about the players, the playing-field in a bottom-up way until we reach scene-level semantic concepts. Each part of our framework is general, so that the system is applicable to several kinds of sports. A central point in our framework is a camera calibration module that relates the *a-priori* information of the geometric layout in the form of a court model to the input image. Exploiting this information, several novel algorithms are proposed, including playing-frame detection, players segmentation and tracking. To address the player-occlusion problem, we model the contour map of the player silhouettes using a nonlinear regression algorithm, which enables to locate the players during the occlusions caused by players in the same team. Additionally, a Bayesian-based classifier helps to recognize predefined key events, where the input is a number of *real-world* visual features. We illustrate the performance and efficiency of the proposed system by evaluating it for a variety of sports videos containing badminton, tennis and volleyball, and we show that our algorithm can operate with more than 91% feature detection accuracy and 90% event detection.**

***Index Terms*—3-D modeling, content analysis, feature extraction, moving object, multi-level analysis, sports video.**

I. Introduction

In consumer videos, sports video attracts a large audience, so that new applications are emerging, which include sports video indexing [1], [2], augmented reality presentation of sports [3], [4] and content-based sports video compression [5]. For these applications, the understanding of the video content considering the user's interests/requests is a critical issue.

Content understanding of sports video is an active research topic, in which the past research can be roughly divided into four stages. Earlier publications [6]–[8] have only focused on pixel and/or object-level analysis, which segment court lines and/or track the moving players and the ball. Evidently, such systems may not provide the semantic interpretation of a sports game. The second generation of sports-video analysis exploits the general features to extract highlights from the sports video. Replayed slow-motion [9], density of scene cuts and sound energy [10] are common input features used by such systems. Although this highlight-based analysis is more general than other proposals, its drawback lies in the insufficient understanding of a sports game, as a viewer cannot deduce the whole story from looking to a special event only. The third stage of sports analysis is an event-based system [1], [2], [5], [11], [12], aiming at extracting predefined events in a specific sports genre. Visual features in the image domain, such as object color, texture and position, are useful clues broadly adopted by these systems. Despite these approaches yield acceptable results within the targeted domain, it is hard to extend the approach of one sports type to another, or even to different matches of the same sports type. In the fourth stage of sports analysis, research shows an increasing interest for constructing a generic framework for sports-video analysis. Opposite to the highlight-based system [9], [10], the new systems [13]–[16] try to recognize more events with rich content by modeling the structure of sports games. A predefined event can be identified based on the model generated during a training phase, which studies the interleaving relations of different dominant scene classes. This type of approaches has been proven to be applicable to multiple sports genres. Unfortunately, the primary disadvantage is that it does not take the behavior of key objects into account, failing to provide sufficient tactical events.

Among the above sports analysis systems, there are several algorithms that are highly related to our work. Sudhir *et al.* [11] propose a tennis video analysis system approaching a video retrieval application. It detects the court lines and tracks the moving players, then extracts the events, such as base-line rally, based on the relative position between the player and the court lines. Improved work is presented in [2], where the authors further upgrade the court detection and player tracking algorithms. [12] first defines four types of camera view in tennis video, involving global, medium, close-up, and audience shots, and then detects events like first-service failure in terms of the interleaving relations of these four views. In the system described in [14], sports video is characterized by its predictable temporal syntax, recurrent events with consistent features, and a fixed number of views. A combination of domain-knowledge and supervised machine learning techniques is employed to detect the re-current event boundaries. [15] also employs shot-based modeling, but creates a concept of mid-level representation to bridge the gap between low-level features and the semantic shot class. Our earlier work [24] proposes a sports analysis system which was intended to be part of a larger consumer media server having analysis features. However, its player tracking algorithm is not capable of handling occlusions caused by multiple players.

Manuscript received February 29, 2008; revised July 08, 2008. First published September 26, 2008; current version published October 29, 2008. This paper was recommended by Associate Editor S.-F. Chang. This work was supported by the ITEA project Cantata.

J. Han is with the Eindhoven University of Technology, 5600MB Eindhoven, The Netherlands (e-mail: jg.han@tue.nl).

D. Farin was with the Eindhoven University of Technology, 5600MB Eindhoven, The Netherlands. He is now with Robert Bosch GmbH, 31139 Hildesheim, Germany (e-mail: dirk.farin@gmail.com).

P. H. N. de With is with Eindhoven University of Technology, Signal Processing Systems Group, 5600MB Eindhoven, The Netherlands. He is also with CycloMedia Technology, 4180BB, Waardenburg, The Netherlands (e-mail: P.H.N.de.With@tue.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2008.2005611

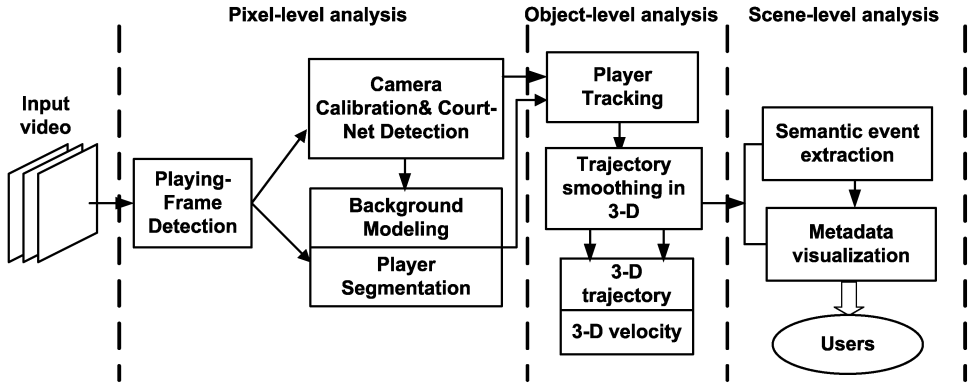


Fig. 1. Architecture of the complete system, which is coarsely separated in the common *pixel-level* and *object-level* analysis, and the application dependent *scene-level* analysis.

Generally speaking, it can be noticed that there are still two problems remaining unsolved, which are indicated below.

1) Unlike most existing systems that only analyze one particular game, the required system should be adapted to more sports games. Note that the generality here does not mean that the system can be exploited to each kind of sports game, since this is very difficult, if not impossible, to achieve. Instead, it might be reasonable and reliable to build a model for the sports having a similar game structure [17], e.g., court-net sports, which includes tennis, badminton, and volleyball.
2) Ideally, a good analysis system should be able to provide a broad range of different analysis results, rather than semantic events only, because of the various requirements from the users. For instance, object-level parameters like the real speed of players, may be helpful to certain users.

In this paper, we present a generic framework for analyzing court-net sports video, intending to address the two challenges mentioned above. Our system is original in three aspects:

- We propose an automatic 3-D camera calibration, which enables to determine the real-world positions of many objects from their image position. Such real-world coordinates are more useful for deducing the semantic events. Additionally, our modeling is generic in the sense that it can be adapted to every court-net sports game, through only changing the court-net layout for each sport.
- We propose several novel pixel- and object-level video processing techniques, where the usage of 3-D modeling is maximized. For example, we construct the background model for player segmentation with the help of the camera calibration, thereby enabling to solve the problem caused by a moving camera.
- We build the entire framework upon the 3-D camera calibration, since this modeling is an efficient tool to link pixel-level analysis, object-level analysis and also scene-level analysis. This framework is advanced due to its capability of providing a wide range of analysis results at different levels, thereby facilitating different applications.

In the sequel, we first give an introduction of our system proposal in Section II, and then describe the proposed 3-D modeling in Section III. Sections IV and V present the algorithms of the pixel-, object- and scene-level analysis. The experimental results are provided in Section VI. Finally, Section VII draws conclusions and addresses our future research.

II. Overview of the Proposed Sports-Video Analysis System

Our sports-video analysis system can be described best as composed of several interacting, but clearly separated modules. Fig. 1 depicts our system architecture with its main functional units and the data-flow. The most important modules are as follows.

1) *Playing-frame detection.* A court-net sports sequence not only includes scenes in which the actual game takes place, but also breaks or advertisements. Since only the playing frames are important for the subsequent processing, we first extract the frames showing court scenes for further analysis.
2) *Court-net detection and camera calibration.* To deduce semantic information from the position and movements of the players, their position has to be known in real-world coordinates. To transform the image coordinates to physical positions, a camera-calibration algorithm has to be applied that uses the lines of the court and net as references to compute the camera parameters.
3) *Player segmentation and tracking in image domain.* The position in the image domain of the player is definitely important to semantic analysis. Here, our basic approach is to segment the location of multiple players in the first frame. Afterwards, we base our player tracking on a popular mean-shift method, but contribute to resolve the practical problems, such as occlusion.
4) *Visual features extraction in the 3-D domain.* The position of each player at each frame is converted to the real-world coordinates using obtained camera matrix. An adaptive Double Exponential Smoothing (DES) filter helps to obtain a more accurate position of each player in real-world coordinates. After that, useful features like trajectory of the player are generated in 3-D domain.
5) *Scene-level content analysis.* This module provides the application-specific semantic information of the sports game that will be important to the user of the system. Obviously, this analysis is depending on the application area of our

framework and cannot easily be generalized. Here, we emphasize on event classification, where we start by modeling events in terms of extracted real-world visual features. Based on this, the combination of machine learning and game-specific contextual information helps to infer the occurrence of a predefined event.

In this paper, the first three modules are denoted as *pixel-level* analysis, since the visual input features used by them are at the pixel level, like color and texture. The forth module intends to investigate the behavior of moving players, so that it is denoted *object-level* analysis. The fifth module aims at extracting the semantically meaningful events and adapting semantic content according to the applications. Thus, it is representing a *scene-level* analysis.

III. 3-D Camera Calibration by Upgrading a Ground-Plane Homography

In this section, we introduce a new algorithm to compute the camera calibration matrix from a single input image of a court-net sports scene. This is different from other approaches that only consider the ground-plane homography [7], [11] and extends our previous work [20], [21]. While the ground-plane homography only establishes the mapping from a 2-D court plane into the 2-D image plane, computing the full camera matrix now also allows us to compute the height of objects if their ground position is known, e.g., the height of players.

A. Camera Calibration Introduction

The task of the camera calibration is to provide a geometric transformation that maps the points in the real-world coordinates to the image domain. Using projective coordinates, we denote a point in image coordinates as $\mathbf{p}' = (u, v, w)^\top$, which corresponds to Euclidean coordinates $(u/w, v/w)$. Similarly, the corresponding point in 3-D space can be written as $\mathbf{p} = (x, y, z, 1)^\top$. According to the projective camera model, the transformation of 3-D points to the image plane can be written as a matrix multiplication with a 3×4 transformation matrix $\mathbf{M}$, transforming a point $\mathbf{p}$ to the image position $\mathbf{p}' = \mathbf{M}\mathbf{p}$:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \quad (1)$$

Since we can freely place the real-world coordinate system, we can choose it such that the ground-plane is placed at $z = 0$. Hence, for the calibration of the ground plane, only 9 significant parameters remain in the matrix $\mathbf{M}$. The reduced matrix is the homography-matrix $\mathbf{H}$ between the ground plane and the image plane

$$\mathbf{H} = \begin{pmatrix} m_{11} & m_{12} & m_{14} \\ m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \end{pmatrix}. \quad (2)$$

Note that $\mathbf{H}$ as well as $\mathbf{M}$ are scaling invariant since a scaling factor is compensated in the conversion to Euclidean coordinates. This leaves eleven free parameters for $\mathbf{M}$ and eight parameters for $\mathbf{H}$. Consequently, $\mathbf{H}$ can be estimated from four point-correspondences in a non-degenerated configuration. In order to estimate $\mathbf{M}$, we first compute $\mathbf{H}$ from four known point-correspondences and then determine the three missing parameters with an additional measurement of the height of the net and the assumption that the camera roll-angle (around the optical axis) is zero (upright camera), which is generally the case.

B. Computing the Ground-Plane Homography

The basic approach of our homography estimation algorithm is to match the configuration of lines in the standard court model with the lines found in the image. The configuration with the best match is selected as the final solution. The intersection points of the lines provide us with the point correspondences to compute $\mathbf{H}$.

The usage of lines instead of using point correspondences directly has the advantage that lines are easy to detect simply by their color and that they can still be extracted even with partial occlusion, e.g., by players. The complete algorithm consists of four stages: line-pixel detection, line-parameter estimation, court model fitting, and model tracking. Each of these four steps is outlined in the following. More details can be found in [20] and [21].

1) Line-Pixel Detection: Detection of white court-line pixels is carried out in two steps. The first step detects white pixels using a simple luminance threshold and the additional constraint on the local structure that prevents that large white areas (like white player clothing) are extracted.

To limit the number of candidate lines to be considered in the initialization step, we apply an additional structure-tensor based filter to remove false detections of court-line pixels in textured areas. The filter criterion is specified such that only linear structures remain.

2) Line-Parameter Estimation: Once we have obtained the set of court-line pixels, we derive parametric equations for the lines. The process is as follows. We start with a RANSAC-like algorithm to detect the dominant line in the data set. The line parameters are further refined with a least-squares approximation and the white pixels along the line segment are removed from the data set. This process is repeated several times until no more relevant lines can be found.

RANSAC is a randomized algorithm that hypothesizes a set of model parameters (in our case the line parameters) and evaluates the quality of the parameters. After several hypotheses have been evaluated, the best one is chosen. More specifically, we hypothesize a line by randomly selecting two court-line pixels $\mathbf{p} = (p_x, p_y)$ and $\mathbf{q} = (q_x, q_y)$. For each line hypothesis, we compute a score $s(\mathbf{g})$ by

$$s(\mathbf{g}) = \sum_{(x',y') \in \mathcal{P}} \max(\tau - d(\mathbf{g}, x', y'), 0) \quad (3)$$

where $d(\mathbf{g}, x, y)$ is the distance of the pixel x, y from line $\mathbf{g}$, $\mathcal{P}$ is the set of court-line pixels and τ is the approximate line width. This score effectively computes the support of a line hypothesis as the number of white pixels close to the line, weighted with their distance to the line. The score and the line parameters are stored and the process is repeated with about 25 randomly generated line hypotheses. Finally, the hypothesis with the highest score is selected, which is illustrated in Fig. 3.

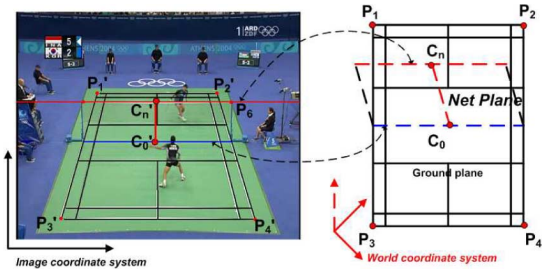
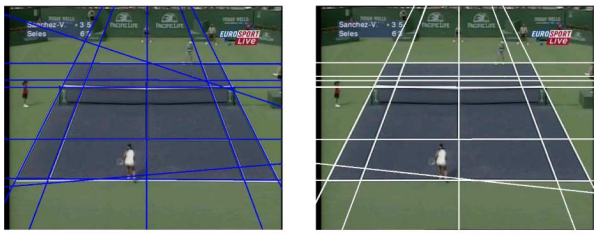


Fig. 2. 3-D modeling: the planes, lines and points are selected in the image and the correspondences in the standard model are determined.



(a) (b)

Fig. 3. Court line detection. (a) Lines detected by RANSAC algorithm. (b) Lines after parameter refinement.

3) Court Model Fitting: The model fitting step determines correspondences between the four detected lines and the lines in the court model. Once these correspondences are known, the homography between real-world coordinates and the image coordinates can be computed. To this end, four intersection points of the lines $\mathbf{p}_i$ and $\mathbf{p}'_i$ are computed (see Fig. 2 for an example), and using the four resulting projection equations $\mathbf{p}'_i = \mathbf{H}\mathbf{p_i}$, eight equations are obtained that can be stacked into an equation system to solve for the parameters of matrix $\mathbf{H}$. Since the correspondences between the lines in the image and the model are not known *a-priori*, we iterate through configurations of two horizontal and two vertical lines in the image as well as in the model. For each configuration, we compute the parameter matrix $\mathbf{H}$ and apply some quick tests to reject impossible configurations with little computational effort. If the homography passed these tests, we compute the overall model matching error E by

$$E = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{M}} \min\left(\|\hat{\mathbf{p}}', \mathbf{H}\mathbf{p}\|_2 + \|\hat{\mathbf{q}}', \mathbf{H}\mathbf{q}\|_2, e_m\right) \quad (4)$$

where $\mathcal{M}$ is the collection of line segments (defined by their two end-points $\mathbf{p}$, $\mathbf{q}$) in the court model and $(\hat{\mathbf{p}}', \hat{\mathbf{q}}')$ is the closest line segment in the image. The metric $\|\cdot,\cdot\|_2$ denotes the Euclidean distance between the two points, and the error for a line segment is bounded by a maximum value e_m. This bound is introduced to avoid a very high error if the input data should contain outliers introduced, e.g., by undetected lines. The transformation $\mathbf{H}$ that gives the minimum error E is selected as the best transformation. Note that this algorithm also works if the intersection point itself is outside the image or if it is occluded by a player.

4) Model Tracking: The previous calibration algorithm only has to be applied in the bootstrapping process when the first frame of a new shot is processed. For subsequent frames, we can assume that the change in camera speed is small. This enables to predict the camera parameters of the next frame with a first-order prediction.[1] Since the prediction provides a good first estimate of the camera parameters, a local search can be applied for refining the camera parameters to match the current view. Let $\mathbf{H_t}$ be the camera parameters for frame t. If we know the camera parameters for frames $t-1$ and t, the transformation between them is $\mathbf{H_{t-1}^{-1}H_t}$, and hence, we can predict the model-to-image parameters $\hat{\mathbf{H}}_{\mathbf{t+1}}$ for $t+1$ by

$$\hat{\mathbf{H}}_{\mathbf{t+1}} = \mathbf{H_t H_{t-1}^{-1} H_t}. \quad (5)$$

The predicted camera parameters have to be adapted to the new input frame. The principle of the parameter refinement is to minimize the distance of the back-projected court model to the white court pixels in the input image. To this end, we use the Levenberg-Marquardt algorithm, which is a fast gradient-ascent algorithm, in order to find the refined transformation.

C. Upgrading the Homography to the Full Camera Matrix

In order to upgrade the homography to a full camera matrix, we make the assumption that a change in object height only affects the vertical coordinate of the image. This implicitly assumes that the camera roll-angle is zero, and that we can neglect the perspective foreshortening in the z-direction (which is the case because the object heights are relatively small compared to the whole field of view). As a consequence, we can write the camera matrix simply as

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} & 0 & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & 0 & m_{34} \end{pmatrix}. \quad (6)$$

Let us now consider the center point of the real-world court. Without loss of generality, we can place the world-coordinate origin $\mathbf{c}_0 = (0,0,0,1)^\top$ to the court center on the ground plane. We denote the height of the net in the court-center as h, which gives the 3-D point $\mathbf{c}_n = (0,0,h,1)^T$. The two corresponding image points are $\mathbf{c}'_0 = \mathbf{M}\mathbf{c}_0 = (m_{14}, m_{24}, m_{34})^\top$ and $\mathbf{c}'_n = \mathbf{M}\mathbf{C}_n = (m_{14}, h\cdot m_{23} + m_{24}, m_{34})^\top$ (in Fig. 2). Finding these two points in the image requires to know the location of net line in the image, which can be detected by using our previous work [22]. Converting to Eucliden coordinates, it follows for the vertical coordinates that

$$y_0 = \frac{m_{24}}{m_{34}} \text{ and } y_n = \frac{m_{24} + h\cdot m_{23}}{m_{34}}, \quad (7)$$

$$y_n = y_0 + h\cdot\frac{m_{23}}{m_{34}}, \quad (8)$$

and finally

$$m_{23} = (y_n - y_0)\frac{m_{34}}{h}. \quad (9)$$

Note that the simplified camera matrix model still models the perspective foreshortening along the real-world x and y axes, and only neglects it along the z-axis. As already mentioned, the depth-variation in the z-axis is small compared to the other dimensions, such that this is a valid approximation.

[1]Note that this does not assume that the camera is moving slowly, but only that the change of speed is smooth.

IV. Pixel- and Object-Level Analysis

This section explains the process to obtain the real-world positions of the player, starting with the detection in the pixel-level, up to the final trajectory computation in the object-level. Since the player detector is active only during the playing frames, we first present our playing-frame detection algorithm.

A. Playing-Frame Detection

Since the subsequent processing steps like object tracking only can give reasonable results when camera calibration information is available, we only use those frames in further processing in which this calibration information is available. These frames are denoted as *playing-frames*.

A trivial method to do the playing-frame detection would be to just use the output of the court-detection algorithm and classify a frame as playing-frame, if a court is detected. However, running the court-detection on a frame that does *not* contain a court can be computationally intensive, especially if the frame contains many white lines (e.g., from the stadium). On the other hand, while we are in the court tracking-step of the camera calibration algorithm, processing is fast and it is easy to observe when the court is disappearing. Hence, we only need a fast algorithm to detect a reappearing court.

Similar to the court-detection step, our playing-frame detection uses only the white pixels of the input frames and can thus reuse the white-pixel detection steps from the court detection. We have found that the number of bright pixels composing the court-net lines is relatively constant over a large interval of frames. This property is reliable for every court-net sports game. Compared to techniques [2], [11], based on the *mean* value of the dominant color, this technique does not require a complex procedure for data training.

Our idea is to count the number of white pixels within the court-area during the court-tracking time-period. When the court disappears, we switch to the court-detection state, in which the previous court-area is observed and the number of white pixels in this area is counted. If this number of white pixels is similar as the number counted during the last playing-frame period, the court-detection algorithm is run again and if successful, a new playing-frame period is started.

In more detail, let $\mathcal{A}$ be the real-world area of the court, extended with a small border (of about 1–2 meters) around the court. Moreover, we denote with $\mathcal{A}'$ the image area obtained by mapping $\mathcal{A}$ into the image space using the previously computed homography $\mathbf{H}$. Finally, the number of white pixels within $\mathcal{A}'$ in frame t is denoted as $F(t)$.

Let t_0 to t_n be the previous time period in which we could track the court. Then, we compute the mean number of white pixels μ_F and the variance of the number of white pixels σ_F. If the court tracking was lost (when we are in the court-detection state), we also compute $F(t)$, but now, instead of updating the statistics, we compare the value to the previously computed mean. If $(F(t) - \mu_F)^2 < 2\sigma_F$, we assume that a court is again visible in the image and the court-detection algorithm is executed. To recover from fatal errors in the statistics, we also compute the court-detection algorithm occasionally, for example once every 500 frames.

B. Moving-Player Segmentation

To analyze a court-net video at a higher semantic level, it is necessary to know the player positions. A class of earlier methods is based on motion detection [11], [27], in which subtraction of consecutive frames is followed by applying a threshold to extract the regions of motion. Another category proposes the exploitation of change-detection algorithms, where the background is first constructed, and subsequently, the foreground objects are found by comparing the background frame with the current video frame. In this paper, our basic idea is to use change detection, but we contribute a novel method to build a background using our court model. In most court-net video sequences, a regular frame (playing-frame) mainly contains three parts: (1) the court (playing-field inside the court lines), (2) the area surrounding the court and, (3) the area of the audience. Normally, the moving area of the players is limited to the field inside the court and partially the surrounding area. This is especially true for the first frame of a new playing-frame shot, where the players are always standing on the baselines. Moreover, the color of the court is uniform, as is also the case for the surrounding area. These features allow us to separately construct background models for the field inside the court and the surrounding area, instead of creating a complete background for the whole image [18]. This approach has two advantages. First, the background picture cannot be influenced by any camera motions. Second, only color and spatial information are considered when we construct the background models, which makes our proposal simpler than methods requiring complex motion-estimation. Basically, our player segmentation algorithm is formed by three steps.

1) Player Segmentation With a Synthetic Background: We model the background as composed of two areas: the court area and its surrounding area. Both can have different colors, as shown in Fig. 4. The location of these two areas in the input frame can be computed directly from the camera calibration information and the preknowledge of the court geometry.

We use the RGB color space for modeling the background, and model the histograms of the individual component by three *Gaussians*. In order to initialize this model, we consider the histograms $H_R(u)$, $H_G(u)$ and $H_B(u)$ of the three color channels, computed in the area within the court-area. The peak of each histogram is the color value most frequently found at each channel, thus, it is expected to be the background. Using a window of width $\pm W$ centered on each histogram distribution ($W = 5$ in our implementation), we compute the mean and variance of the *Gaussian* distribution:

$$\mu_R = \frac{1}{\sum_{u_R^{\max}-W}^{u_R^{\max}+W} H_R(u)} \times \sum_{u_R^{\max}-W}^{u_R^{\max}+W} u H_R(u) \tag{10}$$

$$\sigma_R^2 = \frac{1}{\sum_{u_R^{\max}-W}^{u_R^{\max}+W} H_R(u)} \times \sum_{u^{\max}R-W}^{u_R^{\max}+W} (u-\mu_R)^2 H_R(u). \tag{11}$$

Here, $u_R^{\max}$ represents the R value of the peak point on the histogram map. Equation (10) is for the R channel; the computations for other channels are similar. Note that we compute the

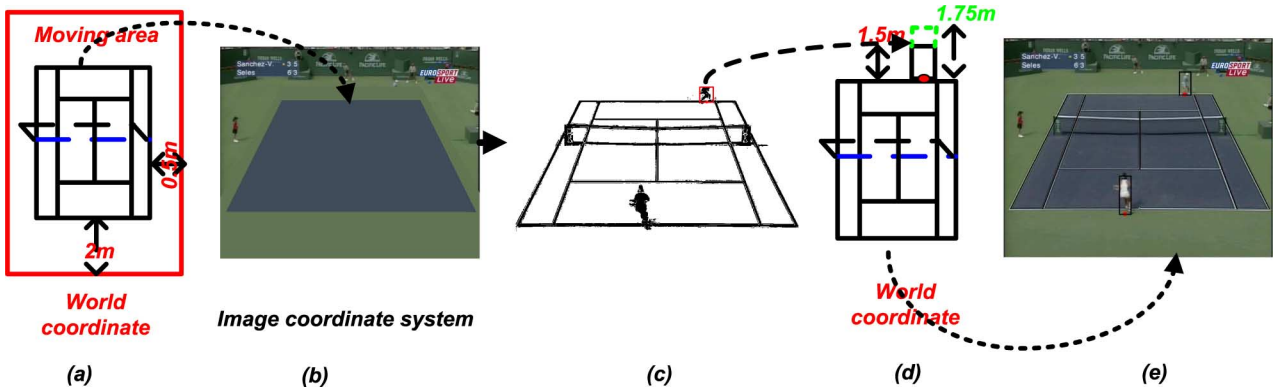


Fig. 4. Player segmentation procedure. (a) Moving areas in the 3-D domain. (b) Generated background model. (c) Background subtraction. (d) We transfer the bounding box in the image to the 3-D domain, and validate the hight of bounding box. (e) We find that the detected bounding box is too small to contain the whole body of the player, so we transform a standard bounding box back to the image and locate the player using it.

mean and variance of each Gaussian distribution based on a reduced range of the histogram, in order to remove the impacts from outliers, such as the player and shadows. We perform the same algorithm to establish the background model for the area surrounding the court field.

Finally, the background model consists if a mean color vector $\boldsymbol{\mu} = (\mu_i)_i$ and the (diagonal) covariance matrix $\boldsymbol{\Sigma} = (\sigma_i)_i$, where these parameters are either from the model within the area of the court or from the model corresponding to the area outside, depending on the pixel position. Based on this color model, we use the Mahalanobis distance $d_k = ((\mathbf{u}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{u}-\boldsymbol{\mu}))^{1/2}$ to perform the foreground/background segmentation, as described in Section IV-B-II.

2) EM-Based Background Subtraction: In our previous method [19], we used a fixed, user-defined threshold to classify the background and foreground pixels. However, our new algorithm computes for each pixel k the posterior probability $p(w_i|d_k)$ with the two classes $(\text{Foreground} = w_1, \text{Background} = w_2)$ and estimates the foreground and background likelihood using the iterative expectation maximization (EM) procedure. More specifically, we compute the posterior $p(w_i|d_k)$ for each pixel and choose the more probable label. Given by the Bayes rule, this posterior equals to

$$p(w_i|d_k) = \frac{p(d_k|w_i,\mu_i,\sigma_i)p(w_i)}{p(d_k)}. \tag{12}$$

Here, $p(d_k) = \sum_{i=1}^{2} p(w_i)p(d_k|w_i,\mu_i,\sigma_i))$, which is represented by a Gaussian Mixture Model (GMM), where $p(d_k|w_i,\mu_i,\sigma_i) = \left(1/\sigma_i\sqrt{2\pi}\right) e^{-\left((d_k-\mu_i)^2/2\sigma_i^2\right)}$. Now, the problem reduces to estimating $p(w_i)$, μ_i and σ_i, which can be iteratively estimated using EM.

The EM process is initialized by choosing class posterior labels based on the observed distance; the larger the Mahalanobis distance of a pixel, the larger the initial posterior probability of being in the foreground:

$$p^{(0)}(w_1|d_k) = \min\left(1.0, \frac{d_k}{255}\right) \tag{13}$$

$$p^{(0)}(w_2|d_k) = 1 - p^{(0)}(w_1|d_k). \tag{14}$$

We found that with this initialization strategy, the process stabilizes fairly quickly, within 10 or so iterations.

3) Player Body Locating: There are several postprocessing steps on the binary map computed by the EM, including shadow-area detection, noisy-region elimination, connected-area labeling, and player-foot position detection [19]. Normally, the whole body of the player close to the camera could be segmented by using our proposal. However, the player far from the camera may not be completely extracted, since some parts, e.g., the head of the player, are out of the surrounding area defined in the background construction step. To address this problem, we rely on our 3-D camera model again. More specifically, we compute the height of the bounding-box containing the segmented body parts of the player in the image domain [see Fig. 4(c)]. Since we detected the foot position of the player in the picture, our 3-D camera model is able to transform this bounding box into the 3-D coordinates and thus to compute its real height. If it is too small, we can deduce that a part of the player's body may not be contained in the bounding box. Therefore, we transform a bounding box with a standard height (man is 185 cm and woman is 175 cm) from the 3-D domain back to the image domain given the foot position of the player. This standard height can also be defined by users, if they know the player's size. By doing so, the complete body of the player could be located in the picture, thereby increasing the robustness of the player tracking algorithm. Fig. 4 portrays the entire procedure of our player segmentation.

C. Multiple Players Tracking and Occlusion Handling

Once we acquired the location of each player in the first frame, it is necessary to track the players over a video sequence. In our system, the mean-shift algorithm [25] is employed, and the player-segmentation results help to refine the foot positions of the players. However, this scheme cannot track the objects through occlusions. Hence, we need to design an occlusion handling algorithm.

Figueroa *et al.* [26] present an automatic player-tracking for soccer games using multiple cameras. They contribute a new method of treating occlusions, which first splits segmented blobs into several player candidates using morphological operators and then employs a backward-forward graph to verify players. However, such a scheme is more suitable to address

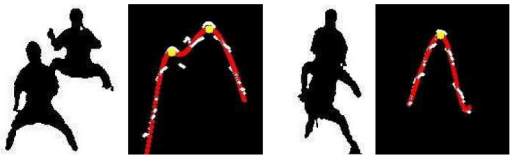


Fig. 5. Blob splitting during the occlusion. Left: binary map. Right: our nonlinear regression result (the dot represents the contour map, the curve is generated by our model, and the top dots represent the detected peak points).

the occlusion among the players in different teams, as the color difference of players is considered in their morphological operators. The frequently occurring occlusion in our application is caused by players in the same team. Our algorithm is also based on two steps: split and verify. But we contribute a splitting method that is performed only on the binary map without color information involved, thereby facilitating to split the blobs in the case of having occlusions in the same team.

We have found that in most court-net games, the occlusion caused by players from the same team is associated with two properties: (1) when looking at the human geometric shape, one should observe a peak in the vicinity of the head, which is true in both the partial and complete occlusion. This phenomenon facilitates to split the blobs. (2) The player usually moves along the direction of the past several frames, and the velocity should not be changed drastically.

1) Blob Splitting: Once acquiring the segmented binary map of players illustrated in Fig. 5, we can obtain the contour of the upper part of the player by using:

$$C(x) = \max\{x | (x, y) \in Q\}, \tag{15}$$

where Q is the collection of the foreground pixels (x, y) on the binary map. After obtaining the contour map, the next step is to find the relative maximum points on it. Instead of directly searching on the map, we search relative maximum points on a *smooth* curve which is automatically generated by the regression technique. Therefore, our method is robust against uncompleted body silhouettes and the noises.

Given the contour $C(x)$, we intend to find a smooth curve (function) to model it. This problem can be solved by minimizing χ^2, which is the sum of the squared residuals over m points for the model $F(\mathbf{c}, x)$, hence

$$\chi^2 = \sum_{i=1}^{m} (C(x_i) - F(\mathbf{c}, x_i))^2. \tag{16}$$

The parameters of the model are collected in the vector $\mathbf{c} = \{c_0, c_1, \ldots\}$. The model $F(\mathbf{c}, x)$ used in this algorithm is a *Gaussian* model, since the geometric shape of the human contour map is similar to the *Gaussian* distribution. Here, the occlusion happens between two players, so that the applied Gaussian model would have two terms, and be written as:

$$F(\mathbf{c}, x) = c_1 \cdot \exp\left(-\frac{(x - c_2)^2}{2c_3^2}\right) + c_4 \cdot \exp\left(-\frac{(x - c_5)^2}{2c_6^2}\right). \tag{17}$$

The Levenberg-Marquardt optimization helps to solve the minimization problem, returning the best-fit vector $\mathbf{c}$. Based on the model, it is feasible to find the relative maximum points on this curve, each corresponding visually to the head of one person. Our algorithm is fully automatic, e.g., it can output two relative maximum points when two persons are partially occluded, but provide only one maximum point when they are completely occluded (see Fig. 5).

2) Player Tracking: When the blobs are split, we need to determine the correspondences between one known player in the previous frame and one blob detected in the current frame. Assume that the known player is T_i and D_j denotes the jth blob candidate extracted in the current frame. Our task is to compute the probability T_i matching D_j, given the velocity and positional data. We estimate the velocity v and its variance σ_v for T_i based on its last M frames (typically 30). Similarly, the motion direction d of T_i can also be modeled by $d \sim N(\mu_d, \sigma_d)$. We model the player motion by

$$P(\tilde{\mathbf{x}} \mid T_i, D_j) = \frac{1}{2\pi|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\tilde{\mathbf{x}} - \boldsymbol{\mu})\right) \tag{18}$$

where $\tilde{\mathbf{x}}$ is the predicted player motion represented by v and d. Parameter $\boldsymbol{\mu} = [\mu_v, \mu_d]$ and $\mathbf{\Sigma}$ is the corresponding covariance matrix derived from σ_v and σ_d.

D. Smoothing Player Motion in the 3-D Domain

In our framework, the semantic-level analysis requires the player position with high accuracy. It is difficult to be provided by only the player extraction and the tracking developed so far. Therefore, we need a procedure that further smoothes and refines the motion of each player. Laviola [23] adopts the DES operator to track moving persons, which executes faster than the Kalman-based predictive tracking algorithm with equivalent prediction performance. Here, we adaptively adjust key parameters of DES filter by using real-world speed of the player calculated by our camera modeling [19].

Once the player's position in the 3-D domain is obtained with high accuracy, the relevant parameters, such as the real speed, trajectory, and so on, can be easily computed. This kind of real-world parameters can be directly provided to the users, like a coach or the player himself. Meanwhile, the semantic-level analysis would also profit from these *real-world* visual features, which will be shown in Section V.

V. Scene-Level Analysis: Bayesian-Based Event Identification

Detection and identification of certain events in a sports game enables the generation of more concise and semantically rich summaries. Our proposal derives several *real-world* visual cues from the previous analysis processes and afterwards performs a *Bayesian* classification for event recognition.

A. Behavior Representation

As mentioned earlier, some existing video analysis systems [5], [18] employ two common visual features: position and speed of the player. In this paper, we first improve these features by computing them in the real-world domain, but we also propose two novel features for event identification, which makes it possible to detect more events. The new features are

speed change (acceleration) and temporal order of the event. In this way, one frame is represented by the feature vector

$$\mathbf{f} = [P_R, S_I, S_C, T_R]. \tag{19}$$

- P_R: this value is actually composed of two parts (P_{R1}, P_{R2}), the relative location between two players and the court field P_{R1} and the horizontal relative relation of two players P_{R2}. More specifically, we divide the court into a net region and a baseline region simply because of their important physical means, and set $P_{R1} = 0$ if both players are in the baseline region and to 1 in case of any one or two payer is in the net region. The value P_{R2} is set to 0 if both players are on the same horizontal half of the court, and 1 if they are on opposite sides.
- S_I: records the average speed of two players.
- S_C: the speed change of the player, i.e., his acceleration status. We use a simple ternary quantization where

$$S_C = \begin{cases} 1, & \text{if both players are accelerating} \\ -1, & \text{if bothplayers are decelerating} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

- T_R: the temporal order of the event. This is defined such that $T_R = 1$ if the current frame belongs to the first $3s$ of a new playing-frame shot; $T_R = 2$ if the current frame belongs to the second $3s$, and so on. We introduced this feature, because we found that in tennis and badminton games, the temporal order of key events is well defined. For example, service is always at the beginning of a playing event, while the base-line rally may interlace with net-approaches.

It should be noted that $\mathbf{f}$ comprises multiple players information. This vector encodes the information of two players from different teams in a single game or, alternatively, encodes the information of two players from the same team in a double game. For the double match, we only analyze the tactical events of the team close to the camera.

B. Event Classification

With the above real-world cues, we intend to model key events of each sports game. We use here two typical events of the tennis game as an explanatory example.

- **Service in a single game:** This event normally starts at the beginning of a playing event, where two players are standing on the opposite half court. In addition, the receiving player has limited motion during the service.
- **Both-net in a double game:** Similar as net-approach in a single game, this is an aggressive tactic in a double match, where both players in a team are volleying nearby the net.

Once the visual features and definitions of the events have been acquired, a classifier is required to label each frame by training pre-defined events. We employ naive Bayesian modeling, which has proved to be popular for classification applications due to its computationally efficient.

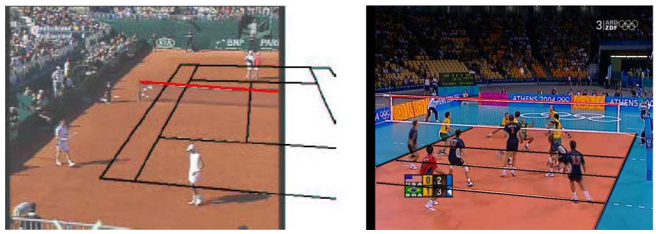


Fig. 6. Detection of the court lines and net line in two challenging cases.

TABLE I
COURT-NET DETECTION AND CAMERA CALIBRATION ACCURACY

Type		Court line	Net line	Calibration
Tennis	visible court	99.1%	98.7%	98.7%
	one line missed	96.3%	95.2%	95.2%
	more lines missed	90.6%	88.2%	86.7%
Badminton		98.7%	96.2%	96.2%
Volleyball		93.4%	88.7%	86.1%

VI. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithms, we have tested our system on more than 3 hours of video sequences that were recorded from regular television broadcasts. Four of them were tennis games on different court classes, three were badminton games including both single and double matches, and two were volleyball games. In our test set, the video sequences have two resolutions: 720×576 and 320×240. Both of them have a frame rate of 25 Hz.

A. 3-D Camera Modeling Technique

We evaluated our 3-D camera modeling with emphasis on the robustness. Robustness is desired due to the fact that in a real broadcast video, the camera is continuously moving and that the video includes full court-views as well as views where a part of the court is occluded or absent in the scene. For our test set, the algorithm is able to find the courts as well as the net line very reliably if the minimum required amount of court lines (two horizontal and two vertical lines) and the net line are visible in the image. Fig. 6 shows some difficult scenes. Table I shows the percentage of successful detections. To highlight the robustness, we calculated the results on three scenarios: the whole court is visible, one court line is invisible, and several lines are invisible. It can be seen that the calibration is correct for more than 90% of the sequences on the average, and our calibration works for more than 85% of the challenging cases. This result is outperforming the algorithms in [2], [11] that only handle full court-view frames. The ground-truth data is manually generated. The most common mis-calibration was caused in the case that the net line is very close to one of the court lines and is mistakenly interpreted as a court line. Furthermore, we have estimated the height of two tennis players (Sanchez and Sales) using our camera modeling technique. The estimated average height of Sanchez was 161 cm in 30 frames, and her real height is 169 cm. Sales' estimated height is 171 cm, and the ground-truth is 178 cm. It can be concluded that our estimation error is less than 5%, whereas the reported error in [28] is 25% (they estimate the height of a goal).

TABLE II
PLAYING-FRAME DETECTION ON TENNIS AND BADMINTON VIDEOS

Game	Method	Total (shots)	Correct	Miss	False
Tennis	Algorithm [11]	148	142	6	21
Tennis	Our algorithm	148	141	7	2
Badminton	Our algorithm	123	115	8	3

B. Results for Pixel and Object-Level Algorithms

In this section, we present the results of our playing-frame detection algorithm, player segmentation and player tracking algorithm. For each of these algorithms, we compare the computed results with manually labeled ground truth data. Since the ground truth of player segmentation and tracking has to be carefully labeled for each frame, it is a time consuming task if we labeled all the videos. Here, we only labeled part of games to demonstrate the performance of our segmentation and tracking algorithms.

1) Playing-Frame Detection: We have conducted the experiments on five complete videos of both tennis and badminton matches. In our dataset, we have in total 148 tennis playing-frame shots and 123 badminton playing-frame shots, each being formed by some playing frames. Additionally, we have implemented the dominant color-based algorithm proposed in [11], and tested it on tennis videos as well. All the parameters are identical to the settings of [11]. Table II lists the comparison results in terms of precision and recall. It can be seen that a promising performance is achieved, leading to $\text{Precision} = 98.04\%$ and $\text{Recall} = 94.39\%$, which is significantly more accurate than the existing technique.

2) Player Segmentation and Tracking in Image Domain: We have applied the described player segmentation algorithm to two difficult cases only, because we have achieved almost errorless results for the case of a fixed, static camera. One difficult case involves a situation where the camera is zooming onto the moving player. Another difficult situation occurs when the camera is rotated in order to follow a moving player. The videos for testing these two cases are manually chosen from the tennis videos in our database. Our algorithm achieves 95% detection rate even in the worst case (see Table III). We have also calculated numerical results of our tracking algorithm for a double-match badminton video clip, dealing with occlusions caused by two or three players. To sum up, our strategy achieves a tracking rate of 97% in the cases without occlusions. Moreover, 85.8% blobs are correctly tracked through occlusions, which clearly outperforms the mean-shift algorithm (only 75.3%).

3) Player Position Adjustment in 3-D Domain: The results discussed here concern a 70-frames clip processed by the smoothing filter in the 3-D domain as described in Section IV-D. Fig. 7 shows examples of the player-positions processed by various smoothing filters, where the results of our adaptive DES compare favorably to the ground-truth data.

C. Results for Scene-Level Analysis Algorithm

We have evaluated our semantic event-classification algorithm on both tennis and badminton broadcast video sequences.

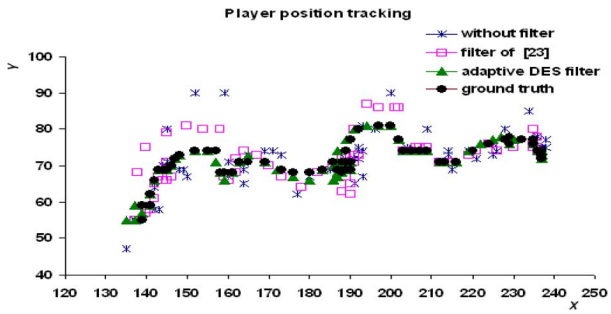


Fig. 7. Player position tracking, using various filtering techniques. X and Y refer to an image domain coordinate system (we track positions in the real-world domain, then transform them back to the image domain).

The total length of the tennis videos is about 100 minutes, ranging from French, US, Australian open to the Olympics 2004. Here, we try to find three typical events for the single match, which are "service", "baseline rally" and "net approach". Similarly, we attempt to detect two events in a double match: "both-baseline" and "both-net", which are typical aggressive and defensive tactics in the tennis double match. The length of badminton videos is about 80 minutes, consisting of two single matches and one double match. Both of them are recorded from the Olympic 2004 event. For the badminton game, we have only extracted the service events. The extraction of other tactical events is also possible through changing the event definitions accordingly.

We have equally divided the video sequences into two parts in terms of the length of the video, where one part was used to training and the other part for testing. All ground truth was labeled manually. Note that we have tested our algorithm only on those frames that successfully completed camera calibration and player tracking components. In order to highlight our 3-D modeling, we have also compared our approach with an alternative technique where image-domain features are used as the input instead of 3-D features. Here, image-domain features are also based on $\mathbf{f}$ in (19), but the parameters are computed by the image domain cues. Table IV summarizes the performance on testing data by using two input types. If we only look at 3-D feature-based algorithm, we can see that the accuracy of the double match is much lower than that of the single match. The main reason is that outputting precise speed of each player is more difficult due to the increase of the number of players. In contrast to image feature-based approach, our algorithm is much better when detecting the "baseline rally" and the "net approach", where the speed and relative position between players and court have a large impact on the computation. Clearly, image-domain features rely on the camera position, which continuously changes from match to match.

D. System Efficiency

In addition to showing the analysis performance of our system, we also want to demonstrate its computational efficiency. In principle, the efficiency of our 3-D modeling technique mainly depends on the image resolution, but is slightly influenced by the content complexity of the picture. To prove it, we calculate the time consumed for each frame when performing our 3-D modeling to a tennis video clip

TABLE III
EVALUATION RESULTS FOR THE PLAYER SEGMENTATION AND TRACKING ALGORITHM

Player	Segmentation when camera is zooming		Segmentation when camera is rotating	
	Correct frames	false frames	Correct frames	false frames
1	2355	85	1578	24
2	2315	125	1562	40

Player	Tracking without occlusion		Tracking during occlusions		
	Tracked frames	missed frames		Solved occlusions	unsolved occlusions
1	2455	9	Our method	678	81
			Mean shift	626	133
2	2201	7	Our method	864	151
			Mean shift	643	372
3	2509	13	Our method	589	112
			Mean shift	521	180
4	2763	18	Our method	365	60
			Mean shift	350	75

TABLE IV
PRECISION AND RECALL CLASSIFICATION RESULTS OF OUR SYSTEM FOR DIFFERENT INPUTS AND SEQUENCES

Type	Event	Feature	Precision	Recall
tennis (single)	service	image	86%	98.1%
		3-D	88.7%	98.5%
tennis (single)	baseline rally	image	76.2%	70.2%
		3-D	87.5%	90.2%
tennis (single)	net approach	image	85.4%	89.1%
		3-D	94.3%	98.5%
tennis (double)	both-baseline	3-D	83.2%	93.7%
tennis (double)	both-net	3-D	81.8%	89.2%
badminton (single)	service	3-D	90.1%	98.2%
badminton (double)	service	3-D	84.3%	90.2%

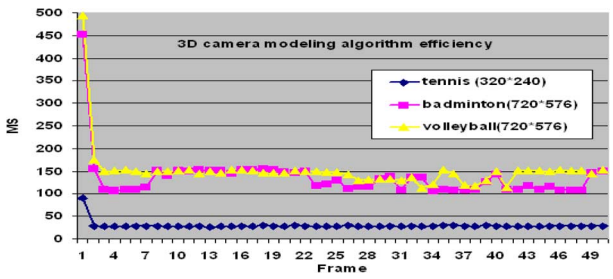


Fig. 8. Execution time of the 3-D camera modeling on a 3-GHz PC.

(320×240), a badminton video clip (720×576) and a volleyball video clip (720×576). The results are given in Fig. 8. For the tennis video, the execution time for the initialization step (first frame) was 90 ms, and the execution times for other frames were between 27 ms and 30 ms, depending on the complexity of the frame. For the badminton and volleyball videos, the initial step required 452 ms and 494 ms, respectively. The average execution times per frame for badminton and volleyball were 128 ms and 144 ms, respectively. The experiments were performed on a P-IV 3-GHz PC. From the results, we can see that the execution time varies with the resolution. For the videos with the same resolution, such as the badminton clip and the volleyball clip in Fig. 8, our algorithm only shows a slight difference. In the volleyball game, there are more line occlusions caused by the moving players.

We have applied the complete analysis system to the single badminton videos (720×576) and calculated the running time of each processing component. The average execution time per frame is around 473.8 ms. Player detection and camera calibration modules use the most computations, which are 64% and 30% of the total execution time, respectively.

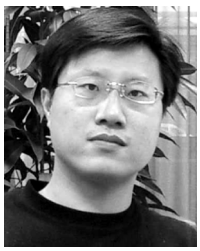
VII. CONCLUSIONS AND FUTURE WORK

In this paper, a general framework for analysis of broadcasted court-net sports video has been introduced. The major contribution of this framework is found in the 3-D camera modeling. At the start, we have computed a homography transformation based on a pose estimation of a court model. Afterwards, the calibration was upgraded to a full camera matrix by adding a measurement of the net height. This enables us to establish a relation between the image domain and the real-world domain. Several novel pixel- and object-level analysis algorithms also profit from this 3-D modeling. Additionally, we have upgraded the feature-extraction part for event classification from the image domain to the real-world domain. It has been proven that this type of visual features is more accurate in the classification. The new algorithms show a detection rate and/or accuracy of 90–98%. At the scene level, the system was able to classify events like service, net approach and baseline rally. A possible enhancement may be the usage of more advanced classifier, such as HMM model, which has been proven to be a powerful model describing the dynamic of the video event.

REFERENCES

[1] Y. Gong, L. Sin, C. Chuan, and H. Zhang, "Automatic parsing of TV programs soccer," in *Proc. IEEE Int. Conf. Mult. Comput. Syst.*, May 1995, pp. 167–174.
[2] C. Calvo, A. Micarelli, and E. Sangineto, "Automatic annotation of tennis video sequences," in *Proc. DAGM Symp.*, 2002, pp. 540–547.
[3] N. Inamoto and H. Saito, "Free viewpoint video synthesis and presentation of sporting events for mixed reality entertainment," in *Proc. ACM ACE*, 2004, vol. 74, pp. 42–50.
[4] J. Han, D. Farin, and P. H. N. de With, "A real-time augmented reality system for sports broadcast video enhancement," in *Proc. ACM Multimedia*, Sep. 2007, pp. 337–340.
[5] S. Chang, D. Zhong, and R. Kumar, "Real-time content-based adaptive streaming of sports videos," in *Proc. Workshop Cont.-Based Acce. Video Libr.*, Dec. 2001, pp. 139–143.
[6] H. Kim, Y. Seo, S. Choi, and K. S. Hong, "Where are the ball and players? soccer game analysis with color-based tracking and image mosaick," in *Proc. Int. Conf. Image Anal. Process.*, Oct. 1997, pp. 196–203.
[7] H. Kim and K. Hong, "Robust image mosaicing of soccer videos using self-calibration and line tracking," *Pattern Anal. Applicat.*, vol. 4, no. 1, pp. 9–19, 2001.
[8] X. Yu, C. Sim, J. Wang, and L. Cheong, "A trejectory-based ball detection and tracking algorithm in broadcast tennis video," in *Proc. IEEE ICIP*, Oct. 2004, pp. 1049–1052.

[9] H. Pan, P. Beek, and M. Sezan, "Detection of slow-motion replay segments in sports video for highlights generation," in *Proc. IEEE ICASSP*, May 2001, pp. 1649–1652.
[10] A. Hanjalic, "Adaptive extraction of highlights from a sport video based on excitement modeling," *IEEE Trans. Multimedia*, vol. 7, pp. 1114–1122, Dec. 2005.
[11] G. Sudhir, C. Lee, and K. Jain, "Automatic classification of tennis video for high-level content-based retrieval," in *Proc. Int. Workshop Cont. Based Acce. Imag. Video Data*, 1998, pp. 81–90.
[12] E. Kijak, L. Oisel, and P. Gros, "Temporal structure analysis of broadcast tennis video using hidden Markov models," in *Proc. SPIE Stor. Retr. Media Data*, Jan. 2003, pp. 289–299.
[13] H. Lu and Y. Tan, "Sports video analysis and structuring," in *Proc. IEEE ICME*, Aug. 2001, pp. 45–50.
[14] D. Zhong and S. Chang, "Structure analysis of sports video using domain models," in *Proc. IEEE ICME*, Aug. 2001, pp. 182–185.
[15] L. Duan, M. Xu, Q. Tian, C. Xu, and J. Jin, "A unfied framework for semantic shot classification in sports video," *IEEE Trans. Multimedia*, vol. 7, pp. 1066–1083, Dec. 2005.
[16] D. Sadlier and N. O'Connor, "Event detction in field sports video using audio-visual features and a support vector machine," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 1225–1233, Oct. 2005.
[17] A. Kokaram, N. Rea, R. Dahyot, A. Tekalp, P. Bouthemy, P. Gros, and I. Sezan, "Browsing sports video: Trends in sports-related indexing and retrieval work," *IEEE Singal Process. Mag.*, vol. 23, no. 2, pp. 47–58, Mar. 2006.
[18] N. Rea, R. Dahyot, and A. Kokaram, "Classification and representation of semantic content in broadcast tennis videos," in *Proc. IEEE ICIP*, Sep. 2005, pp. 1204–1207.
[19] J. Han, D. Farin, and P. H. N. de With, "Multi-level analysis of sports video sequences," in *Proc. SPIE Mult. Cont. Anal., Manage., Retriev.*, Jan. 2006, vol. 6073.
[20] D. Farin, S. Krabbe, W. Effelsberg, and P. H. N. de With, "Robust camera calibration for sport videos using court models," in *Proc. SPIE Stor. Retr. Meth. Appl. Mult.*, Jan. 2004, vol. 5307, pp. 80–91.
[21] D. Farin, J. Han, and P. H. N. de With, "Fast camera-calibration for the analysis of sports sequences," in *Proc. IEEE ICME*, Jul. 2005, pp. 482–485.
[22] J. Han, D. Farin, and P. H. N. de With, "Generic 3-D modelling for content analysis of court-net sports sequences," in *Proc. Int. Conf. Mult. Mode.*, Jan. 2007, pp. 279–288.
[23] J. Laviola, "An experiment comparing double exponential smoothing and Kalman filter-based predictive tracking algorithms," in *Proc. IEEE Int. Conf. Virtual Reality*, Mar. 2003, pp. 283–284.
[24] J. Han and P. H. N. de With, "Unified and efficient framework for court-net sports video analysis using 3-D camera modeling," in *Proc. SPIE Mult. Cont. Acce: Algorithms Syst.*, Jan. 2007, vol. 6506.
[25] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.
[26] P. Figueroa, N. Leite, and R. Barros, "Tracking soccer players aiming their kinematical motion analysis," *Comput. Vis. Image Understand.*, vol. 101, no. 2006, pp. 122–135, 2006.
[27] G. Pingali, Y. Jean, and I. Carlbom, "Real time tracking for enhanced tennis broadcasts," in *Proc. CVPR*, Jun. 1998, pp. 260–265.
[28] Y. Liu, D. Liang, Q. Huang, and W. Gao, "Extracting 3D information from broadcast soccer video," *Image Vis. Comput.*, vol. 24, pp. 1146–1162, 2006.



Jungong Han received the B.S. degree in control and measurement engineering from Xidian University, China, in 1999. In 2004, he received the Ph.D. degree in communication and information engineering from Xidian University.

In 2003, he was a visiting scholar at Internet Media group of Microsoft Research Asia, China, with the topic on scalable video coding. Since 2005, he joined the Department of Signal Processing Systems at the Technical University of Eindhoven, Eindhoven, The Netherlands, where he is leading the research on video content analysis. His research interests are content-based video analysis, video compression and scalable video coding.

Dirk Farin (M'06) graduated in computer science and electrical engineering from the University of Stuttgart, Stuttgart, Germany.

In 1999, he became a Research Assistant at the Department of Circuitry and Simulation at the University of Mannheim, Germany. He joined the Department of Computer Science IV at the same university in 2001 and was a visiting scientist at the Stanford Center for Innovations in Learning in 2003, working on panoramic video visualization. In 2004, he joined the department of signal processing systems at the University of Technology Eindhoven, The Netherlands, and received the Ph.D. degree from this university in 2005. From 2004 to 2007, he supervised the university part of a joint project of Philips and the Technical University of Eindhoven about the development of video capturing and compression systems for 3-D television. In 2008, he joined Robert Bosch GmbH, Corporate Research, Hildesheim, Germany. His research interests include video-object segmentation, 3-D reconstruction, video compression, content analysis and classification.

Mr. Farin received a Best Student Paper Award at the SPIE Visual Communications and Image Processing conference in 2004 for his work on multi-sprites, and two best student paper awards at the Symposium on Information Theory in the Benelux in 2001 and 2003. He organized a special session about sports-video analysis at the IEEE International Conference on Multimedia and Expo in 2005.

Peter H. N. de With (M'81–SM'97–F'07) graduated in electrical engineering from the University of Technology in Eindhoven and received the Ph.D. degree from the University of Technology Delft, The Netherlands, in 1992.

He joined Philips Research Labs Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Department. From 1985 to 1993, he was involved in several European projects on SDTV and HDTV recording. In this period, he contributed as a principal coding expert to the DV standardization for digital camcording. In 1994, he became a member of the TV Systems group at Philips Research Eindhoven, where he was leading the design of advanced programmable video architectures. In 1996, he became senior TV systems architect and in 1997, he was appointed as full professor at the University of Mannheim, Germany, at the faculty Computer Engineering. In Mannheim he was heading the chair on Digital Circuitry and Simulation with the emphasis on video systems. Between 2000 and 2007, he was with LogicaCMG (now Logica) in Eindhoven as a principal consultant. Early 2008, he joined CycloMedia Technology, The Netherlands, as vice-president for video technology. Since 2000, he is professor at the University of Technology Eindhoven, at the faculty of Electrical Engineering and leading a chair on Video Coding and Architectures. He has written and co-authored over 200 papers on video coding, architectures and their realization. He regularly teaches at the Philips Technical Training Centre and for other post-academic courses.

In 1995 and 2000, Mr. de With co-authored papers that received the IEEE CES Transactions Paper Award, and in 2004, the VCIP Best Paper Award. In 1996, he obtained a company Invention Award. In 1997, Philips received the ITVA Award for its contributions to the DV standard. He is a Fellow of the IEEE, program committee member of the IEEE CES, ICIP and VCIP, board member of the IEEE Benelux Chapters for Information Theory and Consumer Electronics, co-editor of the historical book of this community, former scientific board member of LogicaCMG, scientific advisor to Philips Research, and of the Dutch Imaging school ASCII, IEEE ISCE and board member of various working groups.