Robust Camera Calibration for Sport Videos using Court Models

Dirk Farin[a], Susanne Krabbe, Peter H.N. de With[b], Wolfgang Effelsberg[a]

[a]Dept. of Computer Science IV, University of Mannheim, Germany
[b]LogicaCMG / Eindhoven University of Technology, The Netherlands
farin@uni-mannheim.de

ABSTRACT

We propose an automatic camera calibration algorithm for court sports. The obtained camera calibration parameters are required for applications that need to convert positions in the video frame to real-world coordinates or vice versa. Our algorithm uses a model of the arrangement of court lines for calibration. Since the court model can be specified by the user, the algorithm can be applied to a variety of different sports.

The algorithm starts with a model initialization step which locates the court in the image without any user assistance or *a-priori* knowledge about the most probable position. Image pixels are classified as court line pixels if they pass several tests including color and local texture constraints. A Hough transform is applied to extract line elements, forming a set of court line candidates. The subsequent combinatorial search establishes correspondences between lines in the input image and lines from the court model. For the succeeding input frames, an abbreviated calibration algorithm is used, which predicts the camera parameters for the new image and optimizes the parameters using a gradient-descent algorithm.

We have conducted experiments on a variety of sport videos (tennis, volleyball, and goal area sequences of soccer games). Video scenes with considerable difficulties were selected to test the robustness of the algorithm. Results show that the algorithm is very robust to occlusions, partial court views, bad lighting conditions, or shadows.

Keywords: Sport video processing, camera calibration, pose estimation, model-based video analysis.

1. INTRODUCTION AND PREVIOUS WORK

Automatic analysis of sport videos is an interesting application of content analysis, since it allows easy access to the most interesting parts of lengthy sport recordings. This not only offers new convenient features for consumer applications, but also the possibility to analyze a game with automatically deduced game statistics. This will help team coaches to determine strengths and weaknesses of players, or it can be used to entertain the viewer.

For sports played on a court, player positions are semantically important because the player movement or the static player configuration tells much about the current action.[1–5] Clearly, for the analysis of player positions, it is required to know the positions of the players on the court in a real-world coordinate system, rather than the player's position in the image. Hence, the real-world position must be recovered from the image coordinates. For this purpose, the camera calibration parameters have to be estimated from the video input to determine the coordinate system mapping.

Previous work on camera calibration for sport analysis is based on ad-hoc algorithms that were tailored to a specific kind of sport. Sudhir *et al.*[6] describe a calibration algorithm for tennis courts, using a simplified camera model that only considers the camera tilt angle, the camera distance from the court, and the focal-length. Moreover, the algorithm requires that the lower part of the court is non-occluded and a starting position for the search has to be provided. A different approach for tennis court calibration has been proposed by Calvo *et al.*.[2] They apply a Hough transform on the Sobel filter output to find court lines. Assigning the lines to the court model is implemented via a set of heuristics. These impose tight restrictions on the sequences that can be processed. It is assumed, for example, that the two lines at the net are the two lines with the most votes in the Hough transform. But in most tennis videos, the net line is not marked on the court at all. Ekin and Tekalp[3] propose to use a Hough transform to indicate shots that show the goal area in soccer videos. However, no camera calibration parameters are obtained. A camera calibration algorithm for soccer is described by Yamada *et al.*.[4] The camera model includes two rotation axes, focal length, and the camera position. However, the camera position is assumed to be known, which requires tedious manual measurements to be carried out prior to applying the algorithm. The search for the remaining three parameters is carried out using a search over the full parameter space. Especially for the first frame after a cut, when the playing-field location is unknown, this results in high computational cost. Kim and Hong[7] also propose a calibration algorithm for soccer games based on a pan-tilt camera model. The inter-frame transformation is estimated by identifying corresponding lines between frames and using a non-linear approach to determine the homography matrix that minimizes the Euclidean distance between line pairs. Tracking of the camera parameters is used to obtain a good initialization for the parameter optimization in subsequent frames.

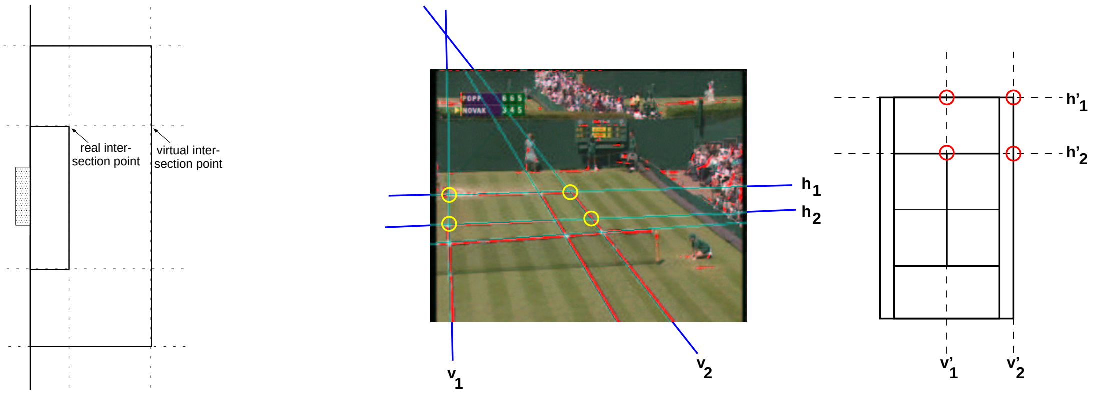
We present a new, more generic camera calibration algorithm that can be adapted to every sport in which the court consists of a sufficient number of straight lines (tennis, football, volleyball, etc.). This configuration of court lines can be specified by the user and integrated into the algorithm as a court model. Based on this model, our algorithm computes the camera parameters for an eight-parameter perspective model. This model can handle arbitrary camera motion, including rotations, translations, and changes of focal length. The algorithm is designed to be robust against cases where large parts of the court are occluded or out of view.

2. ALGORITHM OVERVIEW

Since sport courts can be assumed to be planar, the mapping from the world coordinate system to the image coordinate system can be described by a plane-to-plane mapping[8] (a homography) $\mathbf{H}$. This is an eight-parameter perspective transformation, mapping a position $\mathbf{p}'$ in the model coordinate system to image coordinates $\mathbf{p}$. Writing positions as homogeneous coordinates, the transformation $\mathbf{p} = \mathbf{H}\mathbf{p}'$ equals

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix}.$$

Homogeneous coordinates are scaling invariant, reducing the degrees of freedom for the matrix $\mathbf{H}$ to only eight. In order to determine the eight parameters, at least four point-correspondences between image positions and positions in the court model have to be found. Since courts usually do not have obvious point-features, we use the intersections of lines for establishing point-correspondences. Note that the intersection points themselves do not have to lie inside the image area, since their position can be computed from the line parameters. Moreover, if we represent all visible line segments in the image as infinite lines, we typically get additional *virtual* line intersections that can also be used as calibration points (Fig. 1a).



(a) Real and virtual intersection points.

(b) Line correspondences between image and model.

Figure 1. (a) Real line intersection points occur if the visible line segments cross each other, virtual intersection points can emerge if all line segments are elongated to infinite lines. Our calibration algorithm does not differentiate between real and virtual intersection points. (b) Two horizontal and two vertical lines are selected in the image and the corresponding lines in the model are determined. The pairwise line intersections define four points that are used for calibration.

The basic approach of the algorithm is to extract a number of straight lines from the input image, providing a set of court line candidates. Using a combinatorial search, line candidates are assigned to lines in the court model. For each assignment, the corresponding geometric transformation can be determined. This transformation is used to project the complete court model back to image coordinates. Each transformation is rated by measuring the match between the back-projected model lines and the court lines in the input image. The transformation with the best match is selected as the final solution.

We start with identifying the white pixels of the court lines (Fig. 2, step 1). Because white pixels can also appear on other objects like the players sport dress or advertisment logos, additional constraints are imposed on the court pixel candidates. The initial set of line candidates is determined (step 2) using a Hough-transformation on the previously classified pixels. This transformation finds the parameters of the strongest lines. Moreover, the detection is robust against broken or partially occluded lines. However, in practice, the Hough line detection algorithm also finds lines that are not part of the court, while possibly omitting some lines of the court at the same time. Lines can also be too weak to be discovered, or they can be completely out of the current camera view. For robustness, we therefore do not assume that a predefined set of court-lines will be found in the input image as has been previously assumed in earlier literature.[2, 6] Instead, the court lines that are used for calibration are selected automatically from the set of currently visible lines in the model fitting step.

Court model fitting (step 3) starts with selecting two horizontal and two vertical lines from the set of court-model lines. For each of these lines, a line candidate obtained by the Hough-transform is chosen. By computing the line intersection points in the model as well as in the image, we get four points in the model and corresponding points in the image (Fig 1b). This allows to compute the parameters of our eight-parameters perspective model by solving a linear equation system. Using the obtained geometry transformation, we project the court model onto the input image. The match between the input image and the model is evaluated by counting the number of court line pixels in the image that are covered by the back-projected model. We search for the camera parameters that provide the best match by iterating over all subsets of model lines and line candidates,

In the following frames, the search for the updated court location can be accelerated, since the approximate location of the court is known and a fast local search (step 4) is sufficient to obtain the new camera parameters. Hence, the line detection and model fitting steps are omitted and only the white court-line pixel detector and the camera parameter refinement steps remain (Fig. 2).

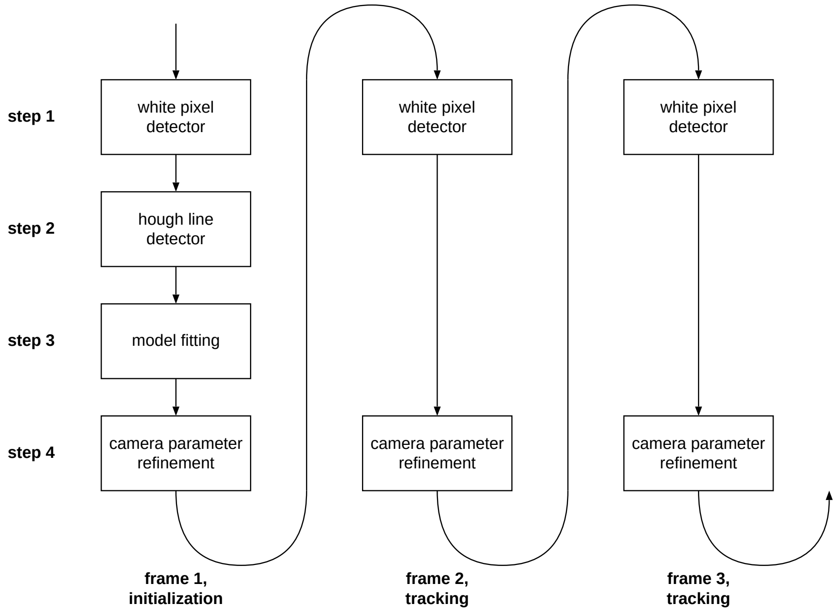


Figure 2. Flowchart of the court-tracking algorithm. At the first frame, the court location has to be initialized. Subsequent frames can use the previous approximate court location, which can be adapted to the new frame using a fast local search.

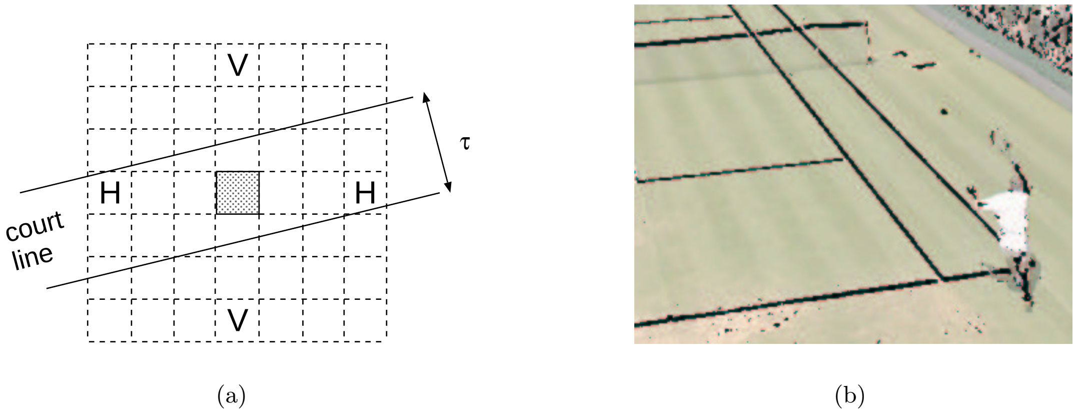
3. COURT MODEL INITIALIZATION

In the remainder of this paper, we assume that input images are represented in YCbCr color space, whereas only the luminance component will be used. We refer to the luminance component of an image as $g(x, y)$.

3.1. White pixel detection

The processing of each frame starts with detecting the pixels of the court lines. In most cases, court lines have a white color. Unfortunately, court lines are usually not the only white objects in the images. Advertisment logos, parts of the stadium, the spectators, or even the players themselves can have white colored parts. Especially in tennis, white is the most common dressing color. If all white pixels were classified as court line pixels, the subsequent Hough line-detection algorithm would create far too many line candidates, thereby making the fitting of the court model time consuming and unreliable. Hence, we need additional criteria to further constrain the set of court line pixels.

Assuming that court lines are typically not wider than τ pixels ($\tau = 8$ in our set-up), we check if the brightness at a distance of τ pixels from the four sides of the candidate pixel are considerably darker than the candidate pixel. Only if they are, the candidate pixel is classified as a white pixel (Figure 3a). Hence, we classify each pixel as court line candidate ($l(x, y) = 1$) or not ($l(x, y) = 0$) according to



(a)

(b)

Figure 3. (a) Schematic, magnified view of part of an input image containing a court line. Each square represents one pixel. The central pixel is only classified as court line pixel if both pixels marked 'H' or both pixels marked 'V' are darker than the central pixel. In the shown case, only the 'V' pixels will be darker. (b) Corresponding white pixel classification result. Note that most of the player pixels are not classified as court line candidates even though the player is dressed in white.

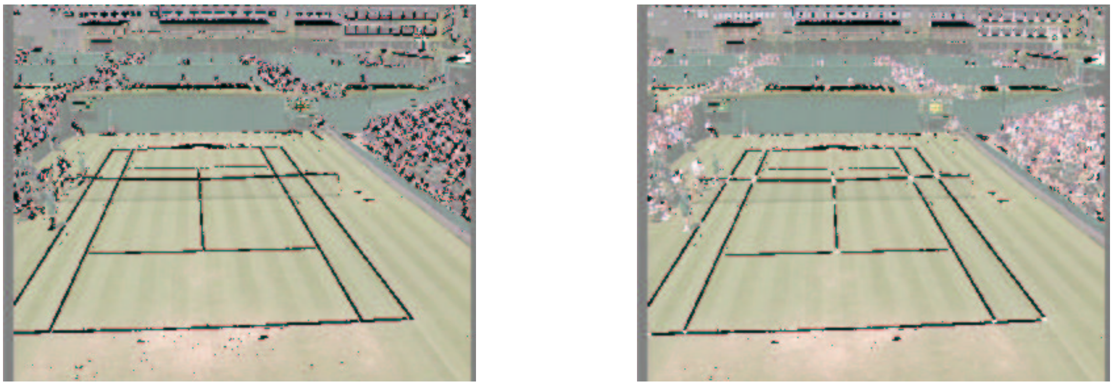
$$l(x,y) = \begin{cases} 1 & g(x,y) \geq \sigma_l \quad \wedge \quad g(x,y) - g(x-\tau,y) > \sigma_d \quad \wedge \quad g(x,y) - g(x+\tau,y) > \sigma_d, \\ 1 & g(x,y) \geq \sigma_l \quad \wedge \quad g(x,y) - g(x,y-\tau) > \sigma_d \quad \wedge \quad g(x,y) - g(x,y+\tau) > \sigma_d, \\ 0 & \text{else.} \end{cases}$$

In the above equation, the first line corresponds to the test if darker pixels can be found at some horizontal distance, assuming that the court line is mostly vertical. The second line performs the analogous test in the vertical direction, assuming that the court line is almost horizontal. The thresholds were set to $\sigma_l = 128$ and $\sigma_d = 20$ in our set-up and worked equally well on a large variety of sequences with different illumination conditions and court colors. Figure 3b shows a sample result of the white pixel detector. It is clearly visible that the additional constraint prevents that the player pixels are also marked as court line candidates despite the player's white clothes.

A further problem can occur with small white letters in logos, white areas in the stadium, or spectators dressed with white clothes, since these pixels in fine textured areas may still pass the above white line test. To prevent that these areas cause too many false detections in the line-extraction step, we exclude white pixels that are in textured regions. Textured regions are recognized by observing the two eigenvalues of the structure matrix $\mathbf{S}$, computed over a small window of size $2b+1$ around each candidate pixel $(p_x; p_y)$. The structure matrix is defined by[9]

$$\mathbf{S} = \sum_{x=p_x-b}^{p_x+b} \sum_{y=p_y-b}^{p_y+b} \nabla g(x,y) \cdot (\nabla g(x,y))^T.$$

If both eigenvalues of matrix $\mathbf{S}$, called λ_1, λ_2 $(\lambda_1 \geq \lambda_2)$ are large, it indicates a two-dimensional texture area. If one eigenvalue is large and the other is small, image gradients are oriented along a common axis. On the straight court lines, the latter case will apply, what we can exploit to define an additional rule which removes white pixels if $\lambda_1 > 4\lambda_2$. Results of the proposed structure constraint can be seen in Fig. 4.



(a) Without line-structure constraint.

(b) Including line-structure constraint.

Figure 4. Detected white pixels with and without applying the line-structure constraint. (a) Many wrong court-line pixel classifications occur in the textured areas. (b) The number of false detections is reduced by adding the constraint that court-line pixel candidates are only allowed if the pixel neighborhood shows a linear structure.

3.2. Court-line candidate detector

3.2.1. Line extraction

A standard Hough transform on the set of the previously detected white pixels is used to detect court-line candidates. The parameter space used for the lines is (θ, d), where θ is the angle between the line normal and the horizontal axis, and d is the distance of the line to the origin. The accumulator matrix is sampled at a resolution of one degree for θ and one pixel for d. Line candidates are determined by extracting the local maxima in the accumulator array that are above a threshold $\sigma_h \cdot W \cdot H$ where W, H are the input image dimensions.

The Hough transform has the disadvantage that thick lines in the input image usually result in a bundle of detected lines, which all lie close together. Another disadvantage of the Hough transform is that the accuracy of the determined line parameters is depending on the resolution of the accumulator matrix. This problem cannot be easily reduced by increasing the accumulator matrix resolution, since this also causes that the inexact parameter samples for an input line spread over a larger area in the accumulator matrix.

We solved both of the above-mentioned problems by introducing a further step after the Hough transform to improve the accuracy of the detected line parameters by computing the best fit line to the input data. Furthermore, lines whose parameters are nearly equal are considered being duplicates and one of them is removed.

3.2.2. Line parameter refinement

The parameters for each detected line are refined by minimizing the distance to the near court-line candidate pixels. Let a line obtained from the Hough transform be parameterized by its normal $\mathbf{n} = (n_x, n_y)^T$ with $||\mathbf{n}|| = 1$ and the distance to the origin d. Consequently, the distance between a point with homogeneous coordinates $\mathbf{p} = (x, y, 1)^T$ and line can be computed with the scalar product $(n_x, n_y, -d) \cdot \mathbf{p}$. This defines the set L of court line candidate pixels that are close to the line as

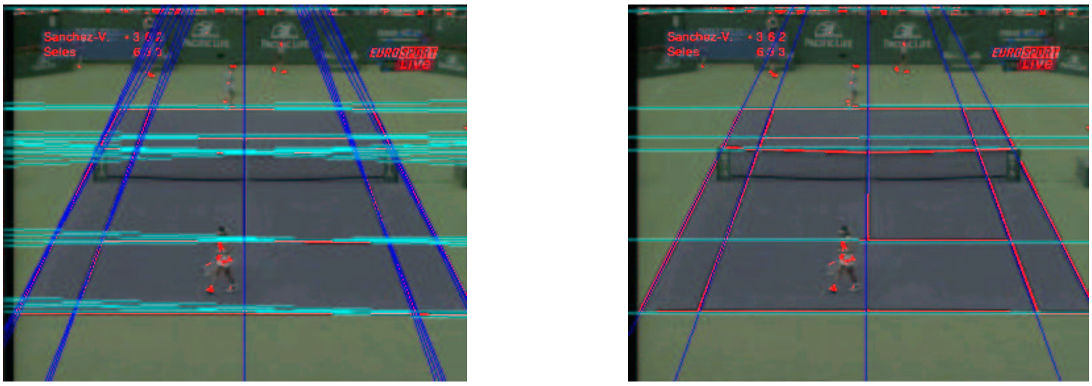
$$L = \left\{ \mathbf{p} = (x\ y\ 1)^T \;\middle|\; l(x,y) = 1 \ \wedge\ |(n_x \quad n_y \quad -d) \cdot \mathbf{p}| < \sigma_r \right\}$$

where σ_r is the maximum distance (e.g., 6 pixels). By stacking all the pixel coordinates in L on top of each other, we obtain a large equation system for the temporary line parameters m_x, m_y. This line representation is used temporarily for the parameter estimation since the above parameterization with homogeneous coordinates is scaling invariant and therefore not directly applicable.

$$\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{|L|} & y_{|L|} \end{pmatrix} \begin{pmatrix} m_x \\ m_y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Since the equation system will be overdetermined in general, we solve it in the least squares sense using a robust LTS estimator.[10–12] The new line parameters are now converted back to the original parameterization by computing $d := 1/\sqrt{m_x^2 + m_y^2}$, $n_x := m_x d$, $n_y := m_y d$. Note that this parameterization does not allow lines that go exactly through the origin. However, this is not a problem since the origin can be shifted far away from the usual court display location.

The new set of lines is scanned for duplicates. Two lines $\mathbf{n_1}, \mathbf{n_2}$ are considered equal if the angle between both is small ($\mathbf{n_1^T n_2} > \cos(0.75°)$) and their distance is also small ($|d_1 - d_2| < 1.5$). The whole refinement and duplicate deletion process is repeated until the number of lines remains stable, which is usually after only three iterations. A sample result is shown in Figure 5.



(a) Lines detected by Hough transform.

(b) Lines after parameter refinement and duplicate removal.

Figure 5. Visualization of court-line candidate extraction. Thick or slightly bent lines in the input lead to a bundle of possible lines (observe, e.g., that the bent net produces many candidates).

3.3. Model fitting

A court model consists of the lines that are drawn onto the ground to define the playfield geometry. The lines are defined in the model coordinate system, which can conveniently be equal to the real-world coordinate system. Remember that the result of our calibration algorithm will define a mapping between the image coordinate system and the model coordinate system, allowing to express player positions in model (i.e., real-world) coordinates. When using the mapping in the opposite direction, we can also mark interesting positions on the court by specifying their real-world coordinates.

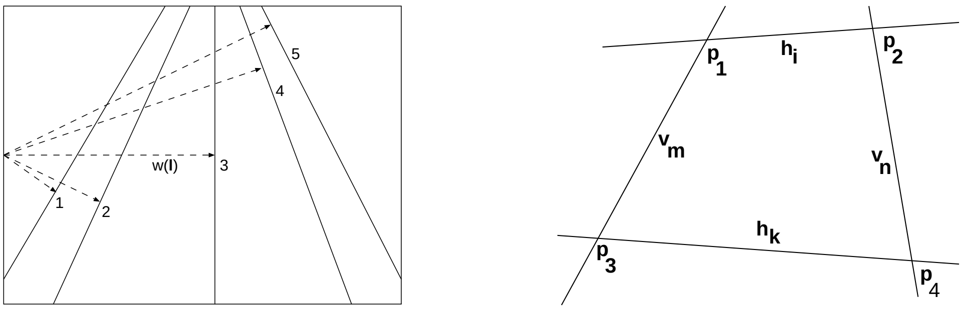
The camera calibration has eight degrees of freedom, which means that we need four corresponding points in the image and in the model. Since prominent points are hard to detect in the input image, we use the intersection of two lines as calibration points. If three or more calibration points are collinear, the camera transformation cannot be determined because the resulting equation system would be rank deficient. Hence, at least two horizontal and two vertical lines are required for proper calibration. The terms *horizontal* and *vertical* here only refer to their position in the court model, since their orientation in the image may be different.

Searching for line correspondences between candidate lines and lines in the model involves a combinatorial search, which is computationally complex. Hence, we introduce further constraints to reduce the search space. First, we classify all lines into sets of horizontal and vertical lines and define an order within the lines. The latter resolves ambiguous assignments of lines with symmetric court models, and it also reduces computational cost since symmetric assignments are only considered once. An additional constraint exploits the fact that the court can never be scaled non-isotropically, i.e., with different scaling factors in the horizontal and the vertical directions.

3.3.1. Finding Line Correspondences

First, all lines in both the image and the court model are sorted into two sets. One set contains all the horizontal lines, while the other set consists of the vertical lines. This classification is trivial for the court-model lines and only has to be done once since the model is predetermined. Candidate lines are classified as vertical if $|\tan^{-1}(n_y/n_x)| < 25^\circ$ and horizontal otherwise. The angle threshold is intentionally made less than 45° because the usual camera configuration is that the camera is located above the ground and looking down. Hence, a general line in the viewing direction will not appear vertical, but a bit slanted because of the perspective projection. The reduced angle threshold accounts for this projection effect. In Figure 5, horizontal lines are drawn in light blue whereas vertical lines are drawn in dark blue.

The set of vertical lines are ordered left to right, the set of horizontal lines top to bottom. Later, when we will search for correspondences between candidate lines and model lines, we will put the constraint on the assignment that the order must be preserved. Sorting the court-model lines is again trivial and can be performed during program initialization. Candidate lines $\mathbf{l} = (n_x, n_y, -d)$ are sorted according to a distance w which we define as $w(\mathbf{l}) = \mathbf{l} \cdot (p_x, p_y, 1)^T$. For vertical lines, we place $(p_x, p_y, 1)$ at the middle of the left border (see Fig. 6a). In the case of horizontal lines, we place $(p_x, p_y, 1)$ at the top border. For further reference, we denote the ordered lines by $\mathbf{h_i}$ for horizontal candidate lines, $\mathbf{v_i}$ for vertical candidate lines, and $\mathbf{h_i'}$, $\mathbf{v_i'}$ for court model lines. Indices are assigned such that $w(\mathbf{h_i}) \geq w(\mathbf{h_{i+1}})$ and similar for the other sets.



(a) (b)

Figure 6. (a) Sorting of vertical lines according to their distance from a point on the left border. (b) Intersection points between horizontal and vertical lines are calculated to obtain four calibration points.

We determine the best line assignment by iterating through all possible assignments. More specifically, we enumerate all pairs of horizontal line candidates $\mathbf{h_i}, \mathbf{h_k}$ with $i < k$, all pairs of horizontal model lines $\mathbf{h'_{i'}}, \mathbf{h'_{k'}}$ and similarly for the vertical lines. Each selected set of lines, either among the candidates or among the model lines, defines a quadrilateral and we calculate its corner points by computing the four intersections (Fig. 6b) according to

$$\mathbf{p_1} = \mathbf{h_i} \times \mathbf{v_m} \ , \quad \mathbf{p_2} = \mathbf{h_i} \times \mathbf{v_n} \ , \quad \mathbf{p_3} = \mathbf{h_k} \times \mathbf{v_m} \ , \quad \mathbf{p_4} = \mathbf{h_k} \times \mathbf{v_n}.$$

The same computation has to be performed on the selected court-model lines, giving the points $\mathbf{p_i'}$. However, since the court model usually only consists of lines parallel to the axes, the intersection computation is trivial.

The calibration parameters for a specific line assignment can be determined by building an equation system based on the perspective camera model and inserting the four point pairs as data:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y_1' & y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2y_2' & y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3y_3' & y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & x_4x_4' & y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & x_4y_4' & y_4y_4' \end{pmatrix} \begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix}.$$

Note that this makes use of the normalization $h_{22} = 1$, which is always true for our application.

3.3.2. Fast Calibration Parameter Rejection Test

If we have C_H horizontal line candidates and C_V vertical candidates, the number of possible input configurations is $\frac{1}{4}C_H(C_H+1)C_V(C_V+1)$. This is quadratic in the total number of candidate lines in the input. Whereas computation of the homography matrix $\mathbf{H}$ for each assignment can be done fast, the evaluation of the model support, which is described in the next Section, is relatively computational intensive. For this reason, we introduced a test that allows to quickly reject physically impossible calibration parameters.

Let us first make the observation that our homography matrix has eight degrees of freedom, but the real-world image formation process has only seven. These comprise three for camera position, three for camera rotation, and one for focal-length. The remaining degree can be attributed to non-isotropic scaling, which refers to unequal scaling in horizontal and vertical directions. If we consider the individual steps of the image formation process, we get

$$\mathbf{p_i} = \mathbf{H}\mathbf{p_i'} = \underbrace{\begin{pmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{pmatrix}}_{\substack{\text{internal camera} \\ \text{parameters}}} \underbrace{\begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{pmatrix}}_{\substack{\text{camera rotation,} \\ \text{translation}}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\substack{\text{non-isotropic} \\ \text{scaling}}} \begin{pmatrix} x' \\ y' \\ z' = 0 \\ 1 \end{pmatrix},$$

where f denotes focal length. Non-isotropic scaling is impossible in the real world. Hence, β should be 1, and we can use this condition as a rejection rule. To determine β from $\mathbf{H}$, we first compensate the camera principal point $(o_x \; o_y)$, which we assume to be at $(W/2 \; H/2)$ by multiplying an appropriate matrix to the left side. Furthermore, we simplify the equations by exploiting the fact that we construct the court model on the $z' = 0$ plane. Consequently, we obtain

$$\begin{pmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & 1 \end{pmatrix} \mathbf{H} = \mathbf{H}' = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & t_x \\ r_{10} & r_{11} & t_y \\ r_{20} & r_{21} & t_z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} fr_{00} & \beta fr_{01} & ft_x \\ fr_{10} & \beta fr_{11} & ft_y \\ r_{20} & \beta r_{21} & t_z \end{pmatrix}.$$

Since the rotation matrix $\{r_{ij}\}$ is known to be orthonormal, we can deduce the two equations

$$\frac{h_{00}'^2}{f^2} + \frac{h_{10}'^2}{f^2} + h_{20}'^2 = \frac{h_{01}'^2}{\beta^2 f^2} + \frac{h_{11}'^2}{\beta^2 f^2} + \frac{h_{21}'^2}{\beta^2}$$

and

$$\frac{h_{00}'h_{01}'}{\beta f^2} + \frac{h_{10}'h_{11}'}{\beta f^2} + \frac{h_{20}'h_{21}'}{\beta} = 0.$$

Finally, we get

$$f^2 = -\frac{h_{00}'h_{01}' + h_{10}'h_{11}'}{h_{20}'h_{21}'} \quad ; \quad \beta^2 = \frac{h_{01}'^2 + h_{11}'^2 + f^2 h_{21}'^2}{h_{00}'^2 + h_{10}'^2 + f^2 h_{20}'^2}.$$

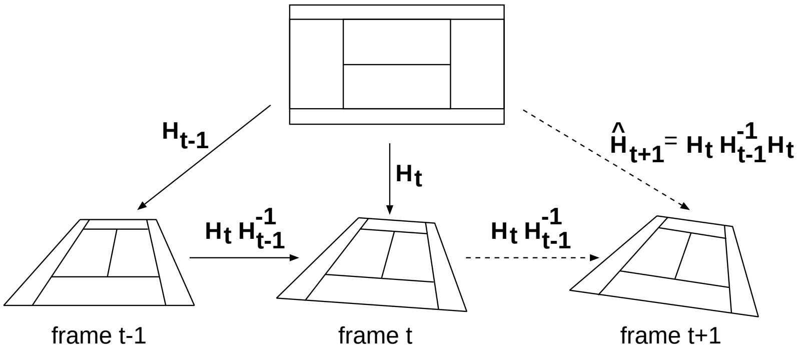


Figure 8. Predicting the camera parameters for frame $t+1$ based on the previously computed parameters for frames $t-1$ and t. The dotted lines indicate predicted values, whereas the solid lines are computed from actual input data.

Because the estimated camera parameters are not perfectly accurate and since the calculation is numerically sensitive, the restriction on β should not be set too tight. We only accept solutions that have $0.5 < \beta < 2$. As an advantageous side effect, this rejection rule not only reduces computation time, but also rejects solutions that physically make no sense.

3.3.3. Evaluating the Model Support

Each set of camera parameters that passed the rejection test is rated by projecting the court model onto the source image and verifying that the model accurately covers the white pixels. The evaluation process transforms all line segments of the model to image coordinates according to the estimated homography matrix $\mathbf{H}$. With $\mathbf{p_i} = \mathbf{H}\mathbf{p}'_\mathbf{i}$, each parameter set is rated by computing the matching score

$$\sum_{\substack{\text{all model line}\\ \text{segments } \mathbf{p}'_\mathbf{i},\, \mathbf{p}'_\mathbf{j}}} \quad \sum_{\substack{\text{all pixels } (x,y)\\ \text{on } \overline{\mathbf{p_i p_j}}}} \begin{cases} 1 & \text{if } l(x,y) = 1, \\ -\frac{1}{2} & \text{if } l(x,y) = 0, \\ 0 & \text{if } (x,y) \text{ is outside of image.} \end{cases}$$

Each model line $\overline{\mathbf{p}'_\mathbf{i}\mathbf{p}'_\mathbf{j}}$ is transformed into the image coordinates $\mathbf{p_i}, \mathbf{p_j}$. This line segment is sampled at discrete positions along the line and the evaluation value is increased by one if the pixel is a white court-line candidate pixel, or decreased by 0.5 if it is not. Parts of the line segment that are not visible are not taken into account (Fig. 7). Non-matching pixels are only penalized with half weight since the detection of court-line candidate pixels is often disturbed by shadows or occlusions.

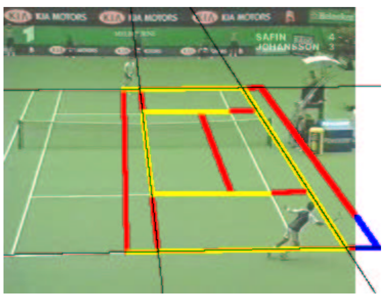


Figure 7. Evaluation of model match. Each pixel that is covered by the proposed model location contributes +1 if the pixel is a court line candidate (marked with *yellow*), −0.5 if it is not (*red*), and 0 if the pixel lies outside the visible image area (*blue*).

After all calibration matrices have been evaluated, the matrix with the largest matching score is selected as the best calibration parameter setting.

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 acceleration of camera motion is small. This enables the prediction of the camera parameters for the next frame. Since the prediction provides a good first estimate of the camera parameters, a simplified version of the above algorithm can be applied.

4.1. Camera parameter prediction

Let $\mathbf{H_t}$ be the camera parameters for frame t. If we know the camera parameters for frames t and $t-1$, we can predict the camera parameters $\hat{\mathbf{H}}_{\mathbf{t+1}}$ for $t+1$ by (see Fig. 8)

$$\hat{\mathbf{H}}_{\mathbf{t+1}} = \mathbf{H_t}\mathbf{H}_{\mathbf{t-1}}^{-1}\mathbf{H_t}.$$

4.2. Camera calibration refinement

The predicted camera parameters have to be adapted to the new input frame. First, white court-line candidate pixels are extracted just as described in Section 3.1. Since we start tracking with a good first estimate of the court location and only a narrow neighborhood is considered, the accuracy of the white-pixel detector can be decreased in favour of faster execution. We therefore disable the final line-structure check during the tracking phase.

In order to update the camera parameters, we determine the inverse of the predicted camera parameter matrix $\mathbf{M} = \hat{\mathbf{H}}_{\mathbf{t+1}}^{\mathbf{-1}}$. This matrix is used to project all pixels that were classified as white into the model coordinate system. Pixels that project to positions too far away from any court line are ignored. The remaining white pixels are grouped with the closest court-model line. We denote white pixels with $\mathbf{p_i} = (p_{i;x}, p_{i;y}, 1)^T$ and their corresponding closest model lines with $\mathbf{l_i} = (n_{i;x}, n_{i;y}, -d_i)^T$. The intention is to find a matrix $\mathbf{M}^\star$ such that the Euclidean distance of all white court pixels to the court model is minimized. The total projection error D is defined as the sum of all pixel distances, thus

$$D = \sum_i \left[\mathbf{l_i^T} P(\mathbf{M p_i})\right]^2,$$

where the operator $P(\cdot)$ normalizes the homogeneous coordinates such that $P : (x\ y\ w) \rightarrow (x/w\ y/w\ 1)$. The non-linear Levenberg-Marquardt minimization algorithm[13] is used to find the $\mathbf{M}^\star$ minimizing D.

5. EXPERIMENTS

We have tested the algorithm on 21 sequences that were recorded from regular television broadcasts. The average sequence length is about 30 minutes. Five of the sequences were soccer games, four were volleyball games, and the remaining twelve were tennis games on different court classes (grass, clay, carpet). All algorithm parameters were kept constant during all experiments, only the correct court model was chosen by the user.

For the soccer video, only goal area scenes could be processed, since the middle part of the soccer field does not contain enough information to do a full camera calibration. However, in this case, camera calibration is only possible if further constraints are imposed on the camera parameters (e.g., no change of focal length, or known camera position).

Figures 9 and 10 show example pictures, whereas some particularly difficult scenes have been selected. Example 10e contains a very large shadow on the court that darkens the image so much that most court-line pixels in the shadow area could not be detected. Nevertheless, the court was detected reliably. In picture 10f, where a superimposed text occludes most of the court, calibration could only be carried out using the two leftmost lines. Thus, the calibration accuracy was not sufficient for the whole court, as can be seen in the lower right part. Because of the large occlusion, the camera parameter refinement step could not correct this small error.

For our test set, the algorithm could find and track the courts very reliably if the minimum required amount of court lines (two horizontal and two vertical lines) were clearly visible in the image. In the most common camera angle which shows an overview of most of the court area, no false calibrations occured with our test set. Even in difficult scenes with strong shadows or large occlusions, the calibration was correct for $> 95\%$ of the sequences. The most common mis-calibration was caused in tennis shots like Fig. 10d where the white line at the top of the net was mistakenly assigned to a court line. The computation time for the camera parameter initialization process mainly depends on the number of line candidates obtained from the Hough transform step. The usual initialization time is approximately 0.5-1 second on a 2 GHz Pentium-IV computer. The court tracking in the subsequent frames runs smoothly in real-time.

6. CONCLUSIONS

In this paper, we have described a new, generic algorithm for camera calibration for sport videos. The algorithm can obtain all eight parameters of a perspective motion model without any user assistance. The geometric model of the court can be adapted to virtually any kind of sport. The adaptability to different kinds of sport using definable court models is a notable improvement of flexibility compared to previously proposed algorithms. In this context, it is also advantageous that the algorithm works reliable without the need to tune any further algorithm parameters. Possible applications for the algorithm exist in systems for the automatic extraction of game statistics, detection of interesting scenes, or automatic game summarization.

The algorithm runs in real-time except the initial model localization, which requires about one second of computation time. We are confident that this time can be further reduced by program optimizations or by applying court model specific heuristics. Possible enhancements of the algorithm would be the inclusion of curved line segments into the court model, which would allow calibration in cases where not enough straight lines are visible (e.g., in the center of soccer fields). Future work will concentrate on using the obtained parameters to determine the real-world coordinates of the players and the ball.

REFERENCES

1. I. D. Reid and A. Zisserman, "Goal-directed video metrology," in *Proc. European Conference on Computer Vision (ECCV)*, pp. 647–658, 1996.
2. C. Calvo, A. Micarelli, and E. Sangineto, "Automatic annotation of tennis video sequences," in *DAGM-Symposium*, pp. 540–547, Springer, 2002.
3. A. Ekin and A. M. Tekalp, "Automatic soccer video analysis and summarization," in *SPIE Storage and Retrieval for Media Databases IV*, pp. 339–350, Jan. 2003.
4. A. Yamada, Y. Shirai, and J. Miura, "Tracking players and a ball in video image sequence and estimating camera parameters for 3D interpretation of soccer games," in *Proc. 16th Int. Conf. on Pattern Recognition*, pp. 303–306, Aug. 2002.
5. Y. Ohno, J. Miura, and Y. Shirai, "Tracking players and estimation of the 3D position of a ball in soccer games," in *Proc. International Conference on Pattern Recognition (ICPR)*, **1**, Sept. 2000.
6. G. Sudhir, J. C. M. Lee, and A. K. Jain, "Automatic classification of tennis video for high-level content-based retrieval," in *IEEE International Workshop on Content Based Access of Image and Video Databases, in conjunction with ICCV'98*, pp. 81–90, 1998.
7. H. Kim and K. Hong, "Robust image mosaicing of soccer videos using self-calibration and line tracking," *Pattern Analysis & Applications* **4**(1), pp. 9–19, 2001.
8. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.
9. B. Jähne, *Digital Image Processing*, Springer Verlag, 2002.
10. P. J. Rousseeuw and K. Van Driessen, "Computing LTS regression for large data sets," *Institute of Mathematical Statistics Bulletin* **27**(6), 1998.
11. J. Clarke, S. Carlsson, and A. Zisserman, "Detecting and tracking linear features efficiently," in *Proc. 7th British Machine Vision Conf. (BMVA), Edinburgh*, R. B. Fisher and E. Trucco, eds., pp. 415–424, 1996.
12. D. M. Mount, N. S. Netanyahu, K. Romanik, R. Silverman, and A. Y. Wu, "A practical approximation algorithm for the LMS line estimator," in *Symposium on Discrete Algorithms*, pp. 473–482, 1997.
13. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C*, Cambridge Univ. Press, 1988.
14. S. Iwase and H. Saito, "Tracking soccer players based on homography among multiple views," in *Visual Communications and Image Processing (VCIP) 2003*, **5150**, pp. 283–292, July 2003.

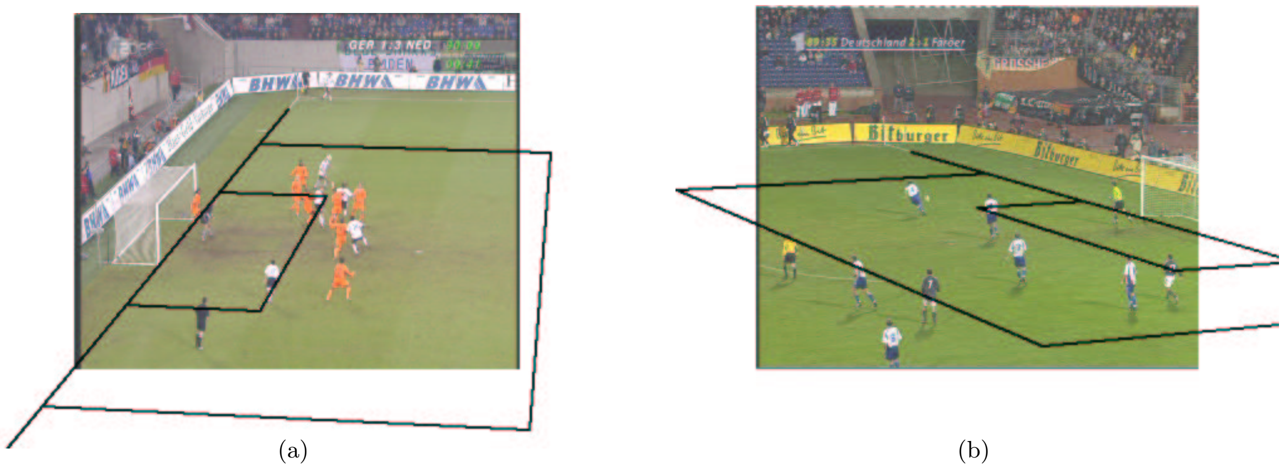
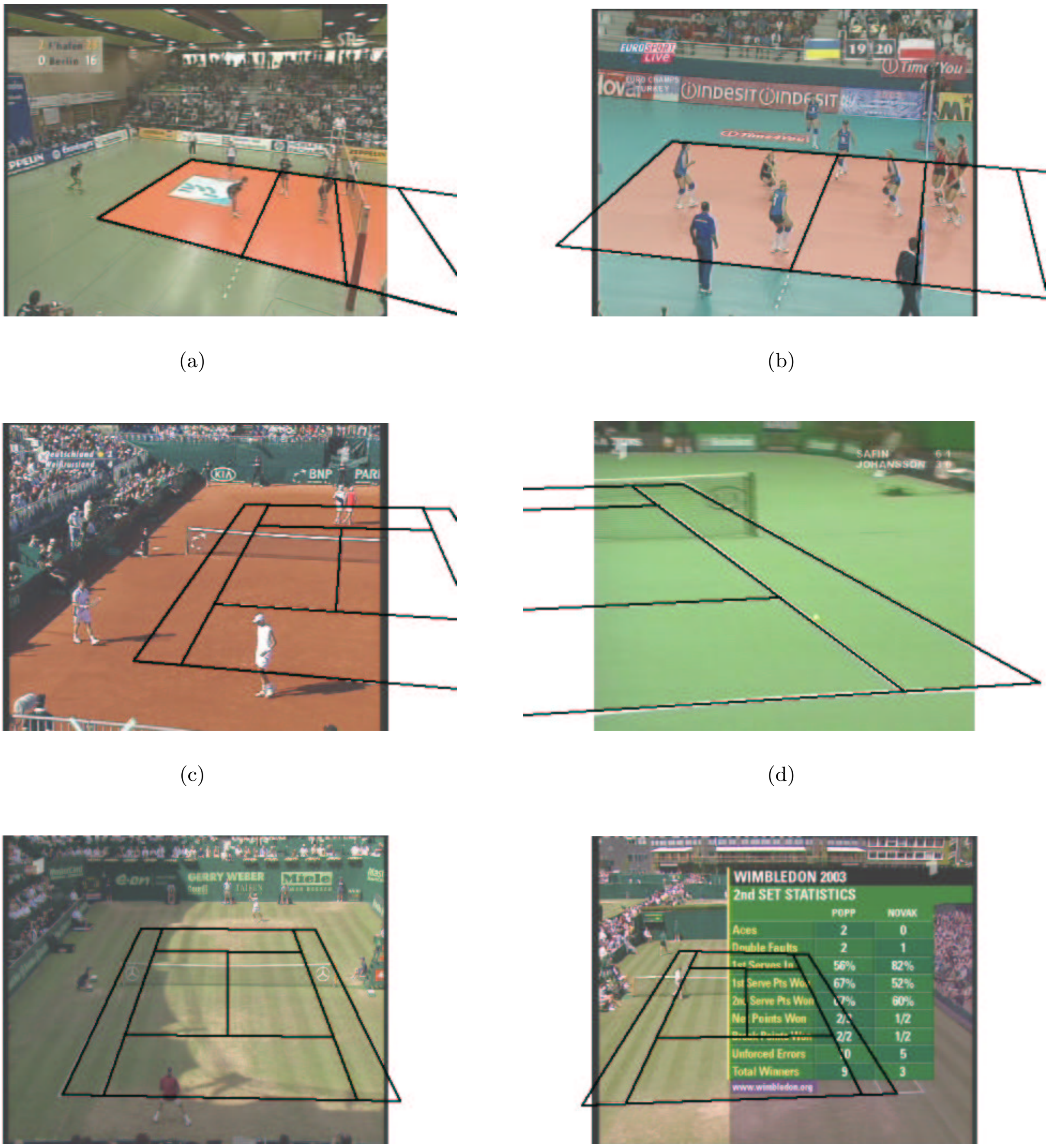


Figure 9. Two soccer scenes.



(a)

(b)

(c)

(d)

(e) scene with a strong shadow

(f) large occlusion

Figure 10. Two volleyball scenes and four examples from different tennis matches.