FAST CAMERA CALIBRATION FOR THE ANALYSIS OF SPORT SEQUENCES

Dirk Farin[1]*, Jungong Han*[1]*, and Peter H. N. de With*[1,2]

[1] Univ. of Technol. Eindhoven, PO Box 513
5600 MB Eindhoven, Netherlands
d.s.farin@tue.nl

[2] LogicaCMG, PO Box 7089
5605 JB Eindhoven, Netherlands
P.H.N.de.With@tue.nl

ABSTRACT

Semantic analysis of sport sequences requires camera calibration to obtain player and ball positions in real-world coordinates. For court sports like tennis, the marker lines on the field can be used to determine the calibration parameters. We propose a *real-time* calibration algorithm that can be applied to all court sports simply by exchanging the court model. The algorithm is based on (1) a specialized court-line detector, (2) a RANSAC-based line parameter estimation, (3) a combinatorial optimization step to localize the court within the set of detected line segments, and (4) an iterative court-model tracking step. Our results show real-time calibration of, e.g., tennis and soccer sequences with a computation time of only about 6 ms per frame.

1. INTRODUCTION

Automatic content analysis of sport videos is an interesting area for computer vision, since it enables new applications like automatic summarization of the highlight scenes of a long sports event, virtual view generation from arbitrary view-points, or computing statistics about the performance or strategy of the players. To analyse the video at a higher semantic level, it is required to know the position of the players or the ball in real-world coordinates of the sports court. Since the playfield is planar, the mapping between real-world coordinates and the observed image can be described with a projective transform. To obtain the transformation parameters, a set of features at well-known positions have to be identified in the image. By establishing correspondences between the detected features and their position on the playfield in real-world coordinates, the transform parameters can be obtained. In the case of sports that have well-defined line structure on the playfield (in the following called *court*), these lines provide a good feature for calibration.

In early work [5], it was proposed to detect four predefined points on a tennis court for calibration. However, the algorithm has to be initialized manually, and it is not robust against occlusions of the court lines connecting these four points. In [4, 6] a more robust detection of the court (for soccer videos) is described, but it requires a computationally complex initialization using an exhaustive search through the parameter space. Calvo *et al.* [1] apply a Hough transformation to detect court-lines, which they subsequently use for calibration. However, they use heuristics to assign the detected lines to lines in the court model that are not applicable to the general case. Finally, it was proposed in [3] to use a combinatorial search to establish correspondences between the lines that were detected with a Hough transform and the court model. This provides a high robustness even for large occlusions or bad lighting conditions. Even though this approach achieves an initialization of the calibration in about one second on a standard 2 GHz Pentium-4 computer, the speed is not sufficient for real-time applications.

If the cameras are fixed, calibration speed is not the main concern, but for the analysis of public broadcasts, camera motion is usually present and calibration parameters have to be determined on-line. In this paper, we will modify the approach of [3] to increase the speed to real-time. The main structure of the system is kept, but most parts are replaced by new, more efficient algorithms. The subsequent section gives an overview of the complete system while Section 3 describes details and modifications of the algorithms used in each of the steps.

2. OVERVIEW OF THE CALIBRATION SYSTEM

The task of a camera calibration system is to provide the geometric transformation that enables to map points in the image to real-world coordinates on the sports court. Since both the court and the displayed image are planar, this mapping is a homography that can be written as a 3×3 transformation matrix $\mathbf{H}$, transforming a point $\mathbf{p} = (x, y, w)^\top$ in real-world coordinates to image coordinates $\mathbf{p}' = (x', y', w')^\top$ as $\mathbf{p}' = \mathbf{H}\mathbf{p}$. Since $\mathbf{H}$ is scaling invariant, eight free parameters have to be determined. They can be calculated from four points whose positions are both known in the court model and in the image. Note that these four points need not be fixed, but can be selected case by case, as some points may be occluded in some views. Instead of using point features directly, we base our calibration algorithm on lines, since detecting the accurate position of a specific point on a court is more difficult than to estimate the position of line segments. Moreover, the detection of lines is more robust since they are hardly ever occluded completely. Starting with a set of lines, we can also compute their intersection points, which then provide good features to determine the calibration parameters. The complete camera calibration system comprises the following algorithm steps.

0-7803-9332-5/05/$20.00 ©2005 IEEE

1. **Court-line pixel detection.** This step identifies the pixels that belong to court lines. Since court lines are usually white, this step is essentially a white pixel detector. The mandatory feature of this step is that white pixels that do not belong to court lines (the player's white clothing, the audience, the stadium) should not be selected.

2. **Line Parameter Estimation.** Starting with the detected white pixels, line parameters are extracted. We apply a RANSAC-based line detector, motivated by [2], which hypothesizes a line using two randomly selected points. If the hypothesis is verified, all points along the line are removed and the algorithm is repeated to extract the remaining dominant lines in the image. In our improved algorithm, we also determine the extent of the line, to obtain line segments instead of infinite lines. Knowing the end points of the lines enables a faster model fitting as only two lines are required for the calibration instead of four.

3. **Model Fitting.** After a set of lines has been extracted from the image, we need to know which line in the image corresponds to which line in the court model. It may also be the case lines are detected other than those present in the model or that some of the lines were not detected. This assignment is obtained with a combinatorial optimization, in which different configurations are tried and evaluated. We provide two combinatorial searches, using either a pair of line segments, or four infinite lines. The first, newly developed, algorithm using line segments is faster, but not applicable to all situations. The second search using infinite lines is more robust to occlusions, but it is slower and it is only used if the first search failed.

4. **Tracking.** When the initial position of the court is known, the computation in successive frames can be carried out more efficiently since model lines can be assigned to the lines that are closest to the predicted position. The tracking step is not explained in this paper, but more information can be found in [3].

At the algorithm start and after shot boundaries, Steps 1-3 are carried out to find the initial location of the court in the first image. For the subsequent frames, only Steps 1 and 4 are applied, since the court position will be close to the old position. Hence, we achieve a high tracking speed since Steps 1 and 4 are computationally cheap.

3. CALIBRATION ALGORITHM DETAILS

3.1. Line Pixel Detection

Detection of white court-line pixels is carried out in two steps. The first step is a simple but fast luminance thresholding scheme with an additional constraint to exclude large white areas from the detection result. A second step excludes white pixels in fine textured areas. Since the second step is computationally expensive, it is only used in the initialization step.

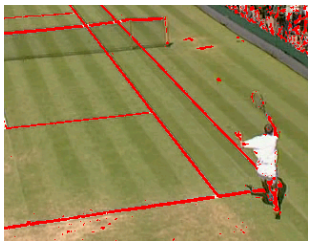


Fig. 1. Court-line pixel detection. Note that the white pixels of the tennis player are not selected.

The first step of the detector puts two conditions on a pixel to be classified as a court-line pixel. Its luminance must exceed a threshold σ_l and either two pixels at a horizontal distance of $\pm\tau$ pixels or at a vertical distance of $\pm\tau$ pixels must be darker than σ_d, where $\sigma_d \ll \sigma_l$. The second condition requires that the pixel must be enclosed either horizontally or vertically by dark pixels. This prevents that white pixels in large white areas are extracted (Fig. 1). The parameter τ should be set to approximately the double court line width.

In the initialization step, no pre-knowledge about the court location is available and we have to consider many alternative placements. To limit the number of candidate lines to be considered in later steps, we apply an additional filter to remove false detections of court-line pixels in textured areas. To this end, we classify the texture in a window w around a pixel with the structure tensor $\mathbf{J} = G_\sigma \star (\nabla f \, \nabla f^\top)$, where ∇f is the image gradient and $G_\sigma \star (\cdot)$ denotes convolution with a Gaussian kernel. Depending on the two Eigenvalues λ_1, λ_2 of $\mathbf{J}$, we can classify the texture into flat (λ_1, λ_2 are small), linear ($\lambda_1 \gg \lambda_2$), and textured (λ_1, λ_2 are large). To reject white pixels in textured areas, we apply the texture test $\lambda_1 > 4\lambda_2$ on all white pixels. Note that this extra calculation is only carried out in the initialization step.

3.2. Line Parameter Estimation

Once we have obtained the set of court-line pixels, we extract 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 start and end position of the line segment are determined and the line-parameters are further refined with a least-squares approximation. Finally, 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. We subsequently explain these steps in more detail.

3.2.1. RANSAC based dominant line detection

RANSAC is a randomized algorithm that hypothesizes a set of model parameters and evaluates the quality of the parameters. After several hypotheses have been evaluated, the

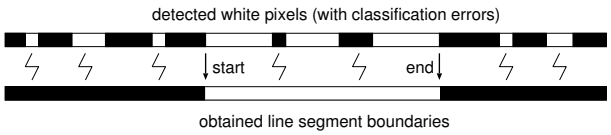


Fig. 2. Detection of line-segment boundaries. At the marked positions, classification errors occurred. The boundaries $start$ and end are placed to minimize the errors.

best one is chosen. Specifically, we hypothesize a line by randomly selecting two court-line pixels, from which we compute line parameters $\mathbf{g}$. For this line hypothesis, we compute a score $s(\mathbf{g})$ as

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

where $\mathcal{P}$ is the set of court-line pixels and $d(\mathbf{g}, x', y')$ denotes the distance between (x', y') and the line. 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 until about 25 hypotheses are generated randomly. At the end, the hypothesis with the highest score is selected.

3.2.2. Line segment boundary detection

Up to now, we obtained the line parameters, but we do not yet know where the line segment starts and where it ends. Since this is valuable information for the subsequent model fitting process, we also compute the line segment boundaries.

Scanning along the detected line, we obtain a sequence of white (court-line) pixels and black (non court-line) pixels. Because of classification errors and occlusions, the data is contamined with noisy data. Assuming that the line segment starts at position $start$ and ends at position end, we define the number of errors as the number of black pixels in the range $start-end$ plus the number of white pixels outside the range (Fig. 2). Using this error definition, we place the line-segment boundaries such that the error is minimized. This optimization has a linear time complexity.

3.3. Model Fitting

The model fitting step determines correspondences between the 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. Searching for the best model requires a combinatorial search that can be computationally complex. Hence, we first try a fast fitting approach that works in most cases, but that is not robust in all cases. If the first approach fails, we determine the model location with a second more robust, but also more complex fitting approach. In the following, we discuss these two fitting approaches and a set of tests that we use to quickly evaluate if a candidate court-model position can be valid.

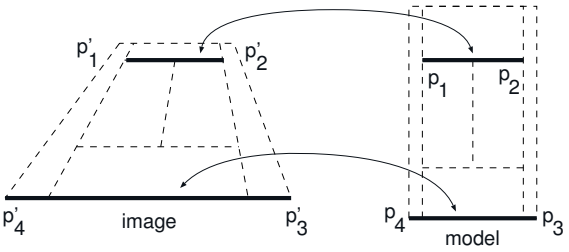


Fig. 3. Fast model fitting. Two corresponding line segments are identified to calculate the transformation parameters from the four end-points.

3.3.1. Fast Fitting

In the fast fitting algorithm, we try to find the transformation parameters by identifying two corresponding line segments in both the image and the model (Fig. 3). To find the best transform, we iterate through all pairs of line segments in the image and in the model. However, we consider only configurations where both lines are parallel, because most configurations of two orthogonal lines have three collinear points, which cannot be used to determine the transformation. For each configuration of lines, we have two end-points for each of the two line segments in both the image and the model. Using these four pairs of points ($\mathbf{p}_i \leftrightarrow \mathbf{p}'_i$), we can determine the homography $\mathbf{H}$.

For each parameter matrix $\mathbf{H}$, we first apply some quick tests to reject impossible configurations with little computational effort (see Section 3.3.3). If the homography passed these tests, we compute the complete model matching error E as

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

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.

The transformation $\mathbf{H}$ that gives the minimum error E is selected as the best transformation. This model fitting algorithm works well if most of the court is visible in the image, as it is mostly the case for tennis broadcasts. For other sports like soccer or volleyball, only small parts of the court are visible at a time and the court lines are clipped at the image boundaries. Obviously, these clipped line segments cannot be used successfully to obtain transformation parameters.

3.3.2. Robust Fitting

If the fast model fitting algorithm does not yield a good solution, we start a robust fitting algorithm that also works with large occlusions and in cases where only a small part of the court is visible. Instead of iterating through possible configurations of two line-segments, we iterate through

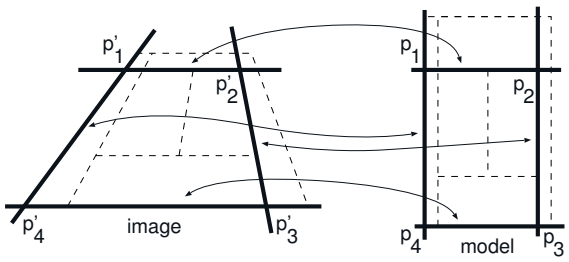


Fig. 4. Robust model fitting. Two horizontal and two vertical lines define four intersection points, which are used to calculate the transformation parameters.

configurations of two horizontal and two vertical lines in the image as well as in the model (Fig. 4). Intersecting the horizontal and vertical lines gives four intersection points, and we can again compute the transformation from these four points. Note that this algorithm also works if the intersection point itself is outside the image or if it is occluded by a player. However, since the number of configurations to be considered is larger, this approach has a higher computational complexity. More details about this step can be found in [3].

3.3.3. Quick rejection tests

Two fast rejection tests are used in the fitting algorithms to quickly reject obviously wrong configurations without the need of computing the full model error.

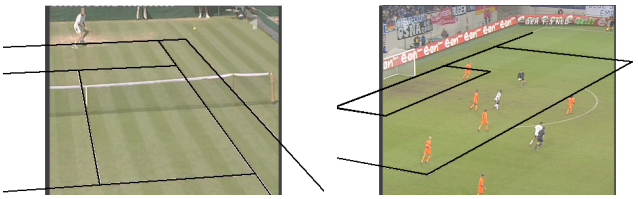
The first test checks the aspect ratio of the court in the image and the court in the model. Because a court is always viewed from a camera at the side of the court, the aspect ratio (vertical extent divided by horizontal extent) is smaller than in the model. We reject a configuration, if the aspect ratio in the image is larger than in the model, or if it is smaller than one fourth of the model aspect ratio.

The second test checks the area of the court in the image. Based on the four corner points of the court, we determine the area of the court outline and reject the transformation if the court area is below one eighth of the image size.

4. RESULTS

We applied the described camera calibration algorithm to several sports sequences showing tennis, soccer, volleyball, and badminton games. Two examples[1] are shown in Figure 5. Note that the algorithm could find the camera calibration for the tennis scene although the view onto the court is very limited. For soccer games, only the goal-mouth scenes were processed, since the number of lines that are visible in the middle of the field are not sufficient to carry out a full eight-parameter calibration. The algorithm was executed on a 2.8 GHz Pentium-4 computer using CIF-resolution input videos. The computation time for the initialization step (first frame) was between 20 ms and 35 ms, depending on the complexity of the frame. Tracking the detected court through the sequence required 4 to 10 ms per frame.

[1]More can be found on `http://vca.ele.tue.nl/demos`.



(a) (b)

Fig. 5. Two example calibration results. The court model has been overlaid with the estimated camera transformation.

5. CONCLUSIONS

This paper described a real-time camera calibration algorithm that can be integrated into an analysis systems for a variety of sports by exchanging the court model. The algorithm comprises a specialized court-line pixel detector, a fast RANSAC-based line segment detection, two court detection algorithms, and a fast tracking mode that is used when the court position has been initialized. All steps of the algorithm have been designed for efficiency, such that the algorithm requires only about 30 ms for the first frame and 6 ms during the court tracking. The algorithm is robust even in difficult scenes with large occlusions or bad lighting conditions, since it adaptively chooses the lines that are used for the calibration process.

6. REFERENCES

[1] C. Calvo, A. Micarelli, and E. Sangineto. Automatic annotation of tennis video sequences. In *DAGM-Symposium*, pages 540–547. Springer, 2002.

[2] J. Clarke, S. Carlsson, and A. Zisserman. Detecting and tracking linear features efficiently. In R. B. Fisher and E. Trucco, editors, *Proc. 7th British Machine Vision Conf. (BMVA), Edinburgh*, pages 415–424, 1996.

[3] D. Farin, S. Krabbe, P. H. N. de With, and W. Effelsberg. Robust camera calibration for sport videos using court models. In *SPIE Storage and Retrieval Methods and Applications for Multimedia*, volume 5307, pages 80–91, 2004.

[4] H. Kim and K. Hong. Robust image mosaicing of soccer videos using self-calibration and line tracking. *Pattern Analysis & Applications*, 4(1):9–19, 2001.

[5] 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*, pages 81–90, 1998.

[6] T. Watanabe, M. Haseyama, and H. Kitajima. A soccer field tracking method with wire frame model from TV images. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 1633–1636, Oct. 2004.