Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

L. Alvarez, L. Gomez, P. Henriquez, J. Sánchez

Real-Time Camera Motion Tracking in Planar View Scenarios

Received: date / Revised: date

Abstract We propose a novel method for real-time camera motion tracking in planar view scenarios. This method relies on the geometry of a tripod, an initial estimation of camera pose for the first video frame, and a primitive tracking procedure. This process uses lines and circles as primitives, which are extracted applying CART (Classification and Regression Tree). We have applied the proposed method to HD (High Definition) videos of soccer matches. Experimental results prove that our proposal can be applied to processing high definition video in real time. We validate the procedure by inserting virtual content in the video sequence.

Keywords Camera motion tracking, Camera calibration, Tripod, Primitives tracking, CART

1 Introduction

Camera motion tracking of video sequences is an important issue in computer vision. It is a challenging problem that involves different techniques, such as feature extraction, feature tracking or camera calibration. It has many applications like, for instance, in 3D scene reconstruction, surveillance applications, broadcast [1], augmented reality [2], virtual reality [3], mosaicking, change of the view point, automatic summarization or virtual objects insertion. Some of these tasks require a highly precise and fast motion tracking of the cameras which are usually mounted on a tripod. These cameras are fixed in location and can freely rotate and change their intrinsic parameters by zooming. Cameras mounted on a tripod have 3 degrees of freedom: Pan, Tilt and Zoom (PTZ). Pan and Tilt represents the tripod axis rotation angles and Zoom represents the lens focal distance.

Most methods for camera motion tracking consist of several stages: initialization, feature extraction (e.g. primitives, key points), movement estimation, tracking and camera parameters computation. We propose a novel method for real-time camera motion tracking in planar view scenarios. An inherent difficulty related to this problem is due to the small number of visible primitives, or due to the large size of HD video sequences. To overcame this problem, we make the following assumptions: the camera is fixed on a tripod, and background (grass) and white lines in the scenario show an appreciable visual contrast in the RGB space. Additionally, we assume that the color of the background presents a rather uniform color.

The main contribution of this paper is the design of a new method to obtain real time camera calibration in planar view scenarios using cameras mounted on a tripod. The proposed method is a combination of the tripod geometry restrictions, a CART method design to extract image primitives and an iterative procedure to camera motion estimation.

We propose an incremental-like approach, that is, we assume small variation between any pair of consecutive frames. This assumption is also reinforced by the fact that the camera motion is highly resctricted due to the tripod. Camera calibration parameters are: extrinsic parameters (3D spatial location) and intrinsic parameters (zoom setting, pixel aspect ratio and the projection of principal point in the image). In our case, these camera parameters are estimated from the PTZ values, which are obtained from information available in the image: the primitives. We denote by primitives the lines and the circles in a model of a soccer field with actual dimensions, which are projected using the homography es-

L. Alvarez · P. Henriquez · J. Sánchez
CTIM: Centro de I+D de Tecnologías de la Imagen.
Departamento de Informática y Sistemas.
Universidad de Las Palmas de Gran Canaria.
Campus de Tafira, 35017, Las Palmas. Spain.
E-mail: (lalvarez, phenriquez, jsanchez)@ctim.es

L. Gomez
CTIM: Centro de I+D de Tecnologías de la Imagen.
Departamento de Ingeniería Electrónica y Automática.
Universidad de Las Palmas de Gran Canaria.
Campus de Tafira, 35017, Las Palmas. Spain.
E-mail: lgomez@ctim.es

timated from the previous two frames. It means that in the primitive tracking stage we search for the primitives in the image using the estimated projection of the reference primitives as starting points. Then we use a decision tree to determine primitive location. In our approach, we use CART [22] to classify primitive pixels and background pixels. Since we start searching the primitive from a near point, always inside the soccer field, we only need to differentiate two classes with the decision tree: primitives (given by white lines or circles) and grass.

This paper is organized as follows: in Sect. 2, we summarize the state of the art. Sect. 3 explains the geometry and calibration process for cameras mounted on a tripod. The proposed method is explained in Sect. 4. We show the experiments and the results in Sect. 5. Finally, in Sect. 6, we present our main conclusions.

2 Related works

Different approaches have been presented to tackle the problem of camera motion tracking in certain situations. A method for self-calibration of rotating and zooming cameras is described in [4], and some other works propose alternative camera motion tracking techniques [5,6]. Other works present methods for multi-camera calibration [7,8]. Based on those techniques, some optimisations for the calibration of PTZ (Pan-Tilt-Zoom) cameras have been introduced [9], and there are also some methods that do not require predefined patterns for tracking [10].

Focusing on sport scenarios, it is common to deal with cameras mounted on a fixed tripod and with a planar view scenario. Therefore, the complexity of the camera pose problem is highly reduced, which greatly simplifies the camera model, as explained in [11]. We can find some works that use the camera motion tracking in this kind of environments [1,12–15]. They follow a similar strategy, which consists of certain tasks applied to each video frame: feature extraction (e.g. primitives and background), camera motion estimation (based on the previous frames), primitive tracking, and improvement of camera parameters estimation.

There exists sensorized tripods that by using inertial systems are able to estimate tripod motion. This kind of methods does not require a visual system but they are expensive and do not provide the high accurate motion estimation required in applications like sport broadcasting.

An important stage of the process is the primitive tracking. Different approaches have previously been proposed to perform the primitive tracking. In [13], a line tracking method guided by camera parameters is described, using a proximity rule to match the detected lines with the estimated ones. A correlation technique is applied in [12]. In [16], camera parameter ranges are defined and a search for the optimal matching is carried out. The approach presented in [17] uses YCbCr color space to detect white lines in the image. Then, the parameters for each line detected are refined by minimizing the distance to the nearest field line. The method described in [15] is based on detecting corresponding feature points between consecutive frames and on object texture extraction method that uses the estimated homography matrix of every frame.

A similar technique for real-time camera tracking in planar view scenarios was presented in [1]. In this case, the proposed method relies on a simplified version of a tripod model: the principal camera point and the tripod rotation center are assumed to be the same and the tripod rotations are modeled using basic rotations around the main axes. The author of this work uses lines as primitives to perform the camera motion matching and applies a variation of the Hough transform to extract the lines in the scene. To initialize the camera position, the author steps through the plausible ranges of pan, tilt and zoom values and projects the lines of the sport court model into the image.

Another method for real-time video calibration focused on soccer scenarios, is described in [18]. The procedure relies on point correspondences of a soccer court model and the extracted lines with the Hough transform, and a KLT (Kanade-Lucas-Tomasi) tracker. The KLT algorithm is necessary when transiting from side views to central views due to the loss of significant lines for homography calculation.

The computation of camera parameters is usually based on a soccer field model using the line correspondece ([1], [17] and [18]). The approach explained in [19] calculates the camera rotation using the vanishing points of the field parallel lines. One variation of the common pitch model is presented in [20], where the authors use the offside lines between the different grass colors, but as there are no rules indicating the number of these lines in a field, they may vary between different stadiums.

In the line detection stage, it is common to proceed as is described in [21]. First of all, the soccer field region is located using a Gaussian Mixture Model. Then, they extract the edges with a Gaussian-Laplacian detector. Finally, to extract the lines, the Hough transform is applied. But, nowadays, decision trees are widely used in image segmentation and classification because they generate easy-to-understand rules and they are fast to train and apply (see, for instance, [23] and [24]). Furthermore, decision trees can manage different features to classify the pixels. This classification is used on images in different areas, for instance, medical images [25] or satellite images [26,27]. They usually work with RGB (Red-Green-Blue) channels combined with other color spaces, such as HSV (Hue-Saturation-Value) or the different bands of satellite images, for example infrared.

3 Geometry and motion of the camera

As it has been explained in Sect. 1, it is usual to find real videos recorded by cameras mounted on a tripod. To be able to track the camera motion in this situation, we have to take into account the mathematical model and geometry related with a tripod. A tripod is defined by a center of rotation $\bar{X}_0 = (X_0, Y_0, Z_0)^T$ and two unitary rotation axes $\bar{e}^0 = \left(\bar{e}^0_x, \bar{e}^0_y, \bar{e}^0_z\right)^T$ and $\bar{e}^1 = \left(\bar{e}^1_x, \bar{e}^1_y, \bar{e}^1_z\right)^T$. The matrix to rotate by an angle of θ_k about axis $\bar{e}^k$, is called $R\left(\bar{e}^k, \theta_k\right)$. In order to rotate a 3D point $\bar{X}$ about axis $\bar{e}^k$ using the center of rotation $\bar{X}_0$, the transformation yields the following equation:

$$\bar{X}\left(\theta_k\right) = \bar{X}_0 + R\left(\bar{e}^k, \theta_k\right)\left(\bar{X} - \bar{X}_0\right). \quad (1)$$

The composition for two rotations of the above type defines the general motion of a tripod. The center of rotation $\bar{X}_0$ is assumed to be the same for both axes. This is equivalent to assume that the two rotating axes of the tripod intersect at a point. The general equation for the motion of a tripod is given by:

$$\bar{X}\left(\theta_0, \theta_1\right) = \bar{X}_0 + R\left(\bar{e}^0, \theta_0\right) R\left(\bar{e}^1, \theta_1\right)\left(\bar{X} - \bar{X}_0\right). \quad (2)$$

From now on, we use the following notation:

$$R\left(\theta_0, \theta_1\right) \equiv R\left(\bar{e}^0, \theta_0\right) R\left(\bar{e}^1, \theta_1\right), \quad (3)$$

$$\bar{t}\left(\theta_0, \theta_1\right) \equiv \bar{X}_0 - R\left(\theta_0, \theta_1\right)\bar{X}_0. \quad (4)$$

Therefore, Equation (2) can be rewritten as:

$$\bar{X}\left(\theta_0, \theta_1\right) = R\left(\theta_0, \theta_1\right)\bar{X} + \bar{t}\left(\theta_0, \theta_1\right). \quad (5)$$

The general equation for the projection of a 3D point $\bar{X} = (X, Y, Z)^T$ onto the image plane is as follows:

$$s\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A\left(f_0\right) R_0 [Id| - \bar{c}^0] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (6)$$

where s is an arbitrary scale factor, $\bar{c}^0 = \left(\bar{c}^0_x, \bar{c}^0_y, \bar{c}^0_z\right)$ is the camera focus, R_0 is a rotation matrix and

$$A\left(f_0\right) = \begin{pmatrix} f_0 & 0 & x_c \\ 0 & rf_0 & y_c \\ 0 & 0 & 1 \end{pmatrix}, \left[Id| - \bar{c}^0\right] = \begin{pmatrix} 1 & 0 & 0 & -\bar{c}^0_x \\ 0 & 1 & 0 & -\bar{c}^0_y \\ 0 & 0 & 1 & -\bar{c}^0_z \end{pmatrix}, \quad (7)$$

with (x_c, y_c) the coordinates of the principal point and f_0 the focal length, which varies with the camera zooming.

In Equation (6) it is assumed that the possible lens distortion has been previously corrected. The projection matrix is $P_0 = A\left(f_0\right) R_0 \left[Id| - \bar{c}^0\right]$. For each frame, i, we can determine the projection matrix with the values of $\left(f_i, \theta^i_0, \theta^i_1\right)$ as follows:

$$P\left(f_i, \theta^i_0, \theta^i_1\right) = A\left(f_i\right) R_0 \left[Id| - \bar{c}^0\right] \begin{pmatrix} R\left(\theta^i_0, \theta^i_1\right) & \bar{t}\left(\theta^i_0, \theta^i_1\right) \\ 0 & 1 \end{pmatrix}. \quad (8)$$

Therefore, considering the following expression:

$$P_i\left(f_i, \theta^i_0, \theta^i_1\right) = A\left(f_i\right) R_0 R\left(\theta^i_0, \theta^i_1\right) \left[Id|R^T\left(\theta^i_0, \theta^i_1\right)\left(\bar{t}\left(\theta^i_0, \theta^i_1\right) - \bar{c}^0\right)\right], \quad (9)$$

we can deduce that, for the frame i, the rotation and focus of the camera after the motion are:

$$R_i = R_0 R\left(\theta^i_0, \theta^i_1\right), \quad (10)$$

$$\bar{c}^i = -R^T\left(\theta^i_0, \theta^i_1\right)\left(\bar{t}\left(\theta^i_0, \theta^i_1\right) - \bar{c}^0\right). \quad (11)$$

One of the main novelties of this approach is that the tripod rotation center and camera projection center are not supposed to be the same, because assuming that both centers are located at the same point is an usual simplification of the model. Notice that any previous view acquired with the actual position of the tripod constitutes a new reference which can be used to estimate the new camera pose along the video sequence.

4 Camera motion tracking by means of a decision tree

The tracking procedure is divided in three main stages: initialization, primitive tracking and camera motion parameters computation (see Fig. 1). In the first step, we obtain some information, such as the geometrical parameters of the tripod (center of rotation and the initial orientation of the two rotation axis), the training classes for the decision tree, and the camera parameters for the first video sequence frame.

Once the initialization stage is done, as explained in Sects. 4.1 and 4.2, we proceed to track the camera motion for the entire video sequence by means of an incremental approach. To perform this step, we track the camera motion in each frame using the tracking information estimated from the previous frames. Usually, this process is initiated from the first frame, but it can be carried out selecting the last frame and reversing the sequence frame by frame. Note that the optimal case should include a subtask consisting of selecting the frame showing the maximum number of visible primitives. From our experience, we prefer to proceed forwardly which is also the natural way.

Once the tripod has been correctly calibrated, tracking the camera motion in each frame consists of estimating three parameters: pan, tilt and zoom. Therefore, for

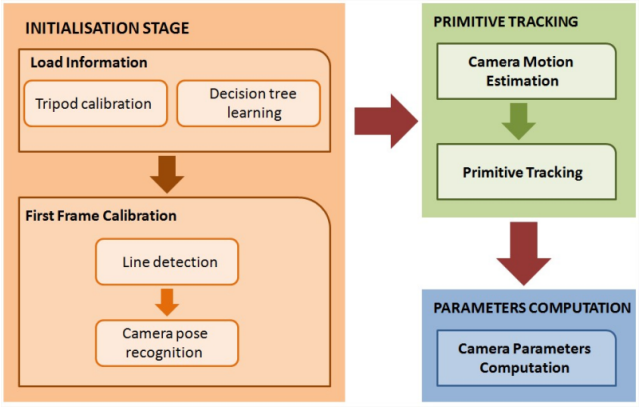

Fig. 1 Camera motion tracking procedure stages.

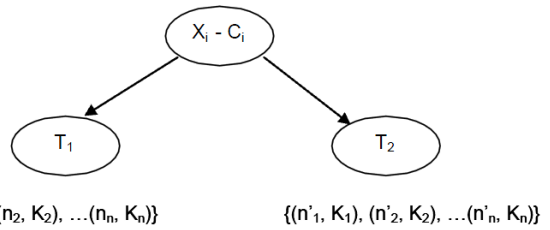

Fig. 2 Binary decision tree. C_i represents the channel we use and X_i the threshold value applied is such channel to obtain the new child nodes T_1 and T_2. n_l represents the number of pixels in node T_1 labeled in the class K_l and n'_l represents the number of pixels in node T_2 labeled in the class K_l.

a given frame n, we have to compute the set p_n, t_n, z_n. Next, we calculate the parameters p_n, t_n, z_n following these steps:

1. Initialize p_n, t_n, z_n. We use a standard extrapolation method to estimate an initial value of p_n, t_n, z_n from the previous frames.
2. Track for the frame n the white lines which delimit the soccer field (explained in Sect. 4.3). Applying a proximity rule, the central points of these lines are associated with the reference field primitives in frame n. The main assumption we do is that the camera moves in a smooth way. Therefore, the central points of the white lines in frame n are close to the previously estimated points.
3. Recompute the parameters p_n, t_n, z_n using the new primitives tracked in frame n. We minimize the RMS error for the field primitives and their recalculated projection with the new camera parameters (explained in Sect. 4.4).

4.1 Tripod and first frame calibration

In order to estimate the geometry of the tripod, we previously calibrate some isolated frames from the video sequence using the automatic camera pose recognition technique explained in [28]. We estimate the geometry of the tripod using a standard bundle adjustment technique.

The last step in the initialization stage is the calibration of the first frame, i.e., to obtain the camera intrinsic and extrinsic parameters. This frame is calibrated using the technique detailed in [28].

This process includes an inner step to calculate the radial lens distortion model for the first frame. This model is used in the whole sequence, but note that it is only calculated once (for the first frame). The distortion model used is the one explained in [29].

4.2 Decision tree building and learning to classify primitives

Building a decision tree is a process that starts with a learning stage. That stage is based on a training set containing information about different classes (see, for instance, [30]). Then, for each video sequence we need a suitable classification data set, which contains information about two classes: primitives and background. In this data set, there are RGB values obtained from a segmentation performed for the first frame of the sequence. Usually, in soccer field scenarios, the primitives are white and the background (grass) is green. From the RGB triplets provided by this training set, a three-channel decision tree is assembled. It determines, within each node, which channel provides the best discrimination between the two classes. With this purpose, we use a measure to estimate the impurity of the sets based on Gini index, which is widely used in CART techniques [31]:

$$\sum_{K \neq K'} P_K P_{K'} = \sum_{K=1}^{N} P_K \left(1 - P_K\right), \tag{12}$$

where P_K is the probability of a point belonging to a class K. In order to decide what channel and threshold are selected at each node, we minimize the resulting impurity measure for the whole set of points (see Fig. 2). The aim is to find the values X_i and C_i that minimize the compound energy in Equation (13), which is calculated from the energies of both child nodes:

$$Gini(X_i, C_i) = E(T_1) + E(T_2), \tag{13}$$

where

$$E(T_1) = \sum_{l=1}^{2} \frac{n_l}{n_1 + n_2} \left(1 - \frac{n_l}{n_1 + n_2}\right), \tag{14}$$

and $E(T_2)$ is defined in a similar way replacing n_l by n'_l (see Fig. 2 for the notation). In our method, this building and learning stage is performed only once for each video

sequence. In the initialization procedure, a single frame is conveniently segmented to provide the two classes: one containing the white delimiting lines and the other containing the grass (see, for example, Fig. 9). Then, with this information, we build and train the decision tree for tracking the primitives. The tree building procedure is explained in Algorithm 1.

```
BuildTree(data)
begin
    if data.K.size = 1 then
        currentNode.K ← data.K;
        return ;          // If only 1 class: leaf.
    else
        D ← FindDiscriminateChannel(data) ;
        // Finds X,C which minimize Eq.(13).
        currentNode.X ← D.X;
        currentNode.C ← D.C;
        S ← SeparateData(data,D.X,D.C);
        leftNode ← BuildTree(S.dl);
        rightNode ← BuildTree(S.dr);
    end
end
```

Algorithm 1: Building CART: *data* is a set composed by n points with an associated K value representing the class and with m values corresponding to each considered channel, *data.K.size* is the number of classes in the set, *S.dl* is a subset extracted from those *data* values in the channel X satisfying $i <= D.C.$, and the subset *S.dr* satisfies $i > D.C.$

4.3 Primitive tracking using the decision tree

Tracking the primitives in the planar view scenarios is required to obtain the position of the objects used for the camera parameters computation. In our case, the elements used to estimate the camera motion tracking are the primitives of a soccer field (lines and circles). After the initialization stage defined in Sects. 4.1 and 4.2, we start the tracking through the video sequence.

For each frame, we first initialize the primitive location using the information of the two previous frames. This initialization provides an estimation of the changes in pan, tilt and zoom values, taking into account the possible camera movement in the previous two frames. These estimated parameters are used to calculate the homography. We project all the points belonging to the reference primitives (lines and circles) onto the frame using this homography. Afterward, we perform an inverse distortion evaluation applying the lens distortion model previously calculated in the initialization stage.

When projecting a reference point onto the image, two cases are possible: the resulting primitive point may be either located on a white primitive or on the background (grass). If the pixel is classified into the primitive

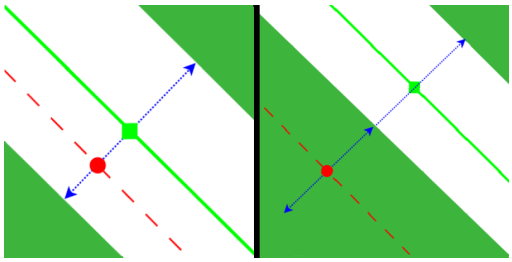

Fig. 3 Search for the edges in the primitive tracking procedure. The red point is the point initialization based on the previous frames. The blue dotted line is the orthogonal line that we examine to find the edges. The green square is the center of the primitive.

class by the decision tree, we search for the edges of the primitive in both directions of the perpendicular orientation (see the left image in Fig. 3). Once we have found the edges, the midpoint between both edges is calculated. This point is considered as the center of the primitive. On the other hand, if the projected point pixel is classified into the background class, we also search for a primitive pixel in both directions of the perpendicular orientation. However, as soon as a primitive pixel is found, we only continue moving in that direction, searching for the other primitive edge. When we find that edge, we calculate the midpoint (see the right image in Fig. 3). The pixels are classified by the decision tree as it is explained in Algorithm 2.

```
Classify(pixel)
begin
    if isLeaf then
        return currentNode.K;
    else
        if currentNode.C(pixel) <= currentNode.X
        then
            return leftNode.Classify(pixel);
        else
            return rightNode.Classify(pixel);
        end
    end
end
```

Algorithm 2: Classifying a pixel: *currentNode.C(pixel)* is the value of the pixel in the channel C, which is the discriminant channel of the node, *currentNode.X* is the threshold and *currentNode.K* is the class.

In both situations, in order to avoid considering large white zones, such as advertisements or players wearing white clothes, the thickness of the primitives has to be controlled by means of a threshold parameter. This threshold is dynamically obtained because farther primitives are thinner than closer ones. We calculate this threshold as the distance between the projections of a reference point and another point obtained with the addition of

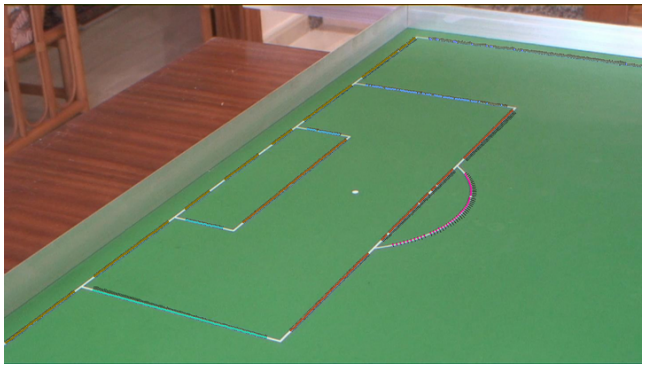

Fig. 4 Primitive tracking stage in a scale model sequence. Black points represent all analyzed points to find the white primitives. Colored points are those selected as primitive center.

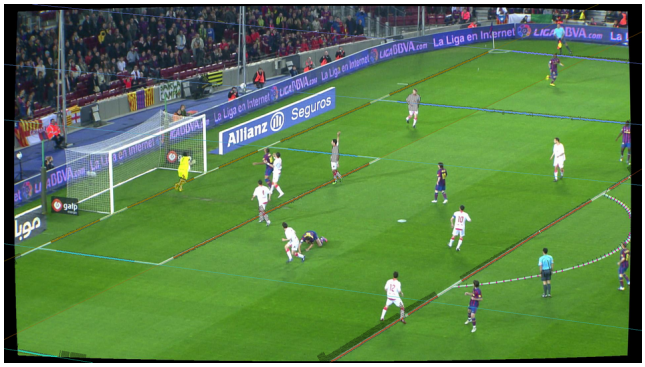

Fig. 5 Primitive tracking stage in a real soccer match video. Black points represent all analyzed points to find the white primitives. Colored points are those selected as primitive center.

a certain thickness (in this case the actual thickness of a soccer field line). If more pixels than the width limit have been examined, the primitive detection is rejected at that point. We can see that by using this threshold parameter it is avoided to consider players occluding the primitives (see Fig. 5).

Instead of processing all the image pixels, as it is done in the method described in [32] and used in our initialization stage, with this tracking proposal we only have to classify pixels in a primitive neighborhood of the previous primitive location. This tracking strategy allows to reduce the processed image pixels to a restricted neighborhood around the primitive location estimations (see Fig. 4 and Fig. 5). Consequently, the processing time is highly reduced compared to the brute-force alternative of processing all the pixels in the image. This feature, added to the fast pixel classification provided by the decision tree, makes it possible to attain real-time processing.

4.4 Camera motion parameters computation

The camera motion parameters computation is the stage where the pan, tilt and zoom parameters (p_n, t_n, z_n) are calculated for the current frame. First, we use camera parameters provided by the previous frames to initialize (p_n, t_n, z_n) using a basic linear extrapolation procedure. Next we optimize (p_n, t_n, z_n) by minimizing the projection error of the obtained image primitive center points to the reference soccer court primitives, that is, we minimize the error function:

$$L(H) = \frac{1}{\mid P \mid} \sum_{p_i \in P} distance(H(p_i), C)^2, \tag{15}$$

where C is the collection of tracked court primitives and P the reference primitives. H is a homography (a 3×3 matrix, obtained from pan, tilt and zoom) and p is a 2D point. We denote by $H(p)$ the perspective transformation induced by H on point p. The function *distance* is the Euclidean distance between the court primitives and their transformations.

Finding the minima of the above function is a difficult problem. The method we propose is based on building homography candidates from the estimated parameters. We observe that as the admissible camera maneuvers is highly constrained, the number of candidates to deal with is small and so is the number of iterations to reach a suitable solution. After significantly reducing the space of search, the well-known Levenberg-Marquardt algorithm is applied.

As we have an approximation of the homography, we can obtain the camera parameters in situations where the standard procedures would fail. For example in images without enough visible primitives to perform the matching with the corresponding reference lines or circles.

5 Experiments and results

5.1 Decision tree configuration experiments

In order to achieve the best results for classification, we have tested different CART configurations varying the number of channels. These tests consist in building a decision tree with a set of training data. After the learning procedure, the data is classified with the decision tree and the pixels classified with errors are counted. Finally, we compare decision tree classifications with manual classifications on four random frames extracted from each video showing different parts of the soccer field. The results are shown in Table 1 and Table 2, where the first row shows the classification results of training data set and other rows are the classification results for some random video sequence images. We can observe that the best configuration for the decision tree channels is RGB (it has the lowest average classification error rate).

The performance of the decision tree is also illustrated in Fig. 6 and Fig. 7. We note that we only focus in the grass/primitive discrimination in a neighborhood of

Table 1 Decision tree configurations percentage error rates, scale model images (best values in boldface).

R	G	B	H	S	V	RGB	HSV
0.068	0.268	0.153	1.112	0.060	0.268	**0.005**	0.009
0.067	0.128	0.100	0.563	0.055	0.128	**0.049**	0.051
0.087	0.119	0.101	0.399	**0.065**	0.119	0.066	0.072
0.069	0.110	0.067	0.270	0.076	0.110	**0.059**	0.046
0.049	0.049	0.076	0.475	**0.035**	0.091	0.040	0.047

Table 2 Decision tree configurations percentage error rates, real images (best values in boldface).

R	G	B	H	S	V	RGB	HSV
0.200	0.344	0.194	0.398	0.322	0.344	**0.067**	0.070
0.174	0.246	0.157	0.297	0.230	0.246	**0.121**	0.124
0.231	0.358	0.201	0.414	0.322	0.358	**0.102**	0.105
0.215	0.268	0.208	0.396	0.315	0.268	**0.163**	0.166
0.146	0.292	0.154	0.263	0.206	0.292	**0.124**	0.134

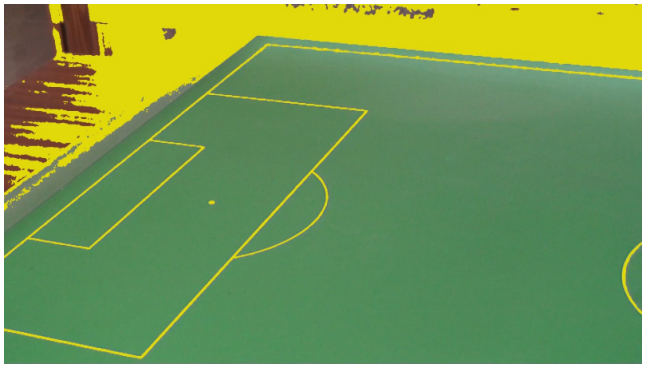

Fig. 6 Scale model image segmented by means of the CART method. The points classified as primitive are painted in yellow.

Fig. 7 Real image segmented by means of the CART method. The points classified as primitive are painted in yellow.

primitive initialization so, the result of the decision tree for other areas in the image is not relevant.

We have compared the classification results obtained with the decision tree using RGB and the results of classifying with the morphological method described in [32]. We have tested both methods with four manually segmented images from two different videos, scale model and real soccer match videos. In Table 3 the percentage error rates are shown.

Table 3 Comparison of percentage error rates between decision tree method and morphological method. RGBs and RGBr are results obtained with CART in scale model images and real images. Ms and Mr account for the results using the morphological method.

RGBs	Ms	RGBr	Mr
0.049	0.072	0.121	0.198
0.066	0.253	0.102	0.312
0.059	0.259	0.163	0.328
0.040	0.084	0.124	0.185

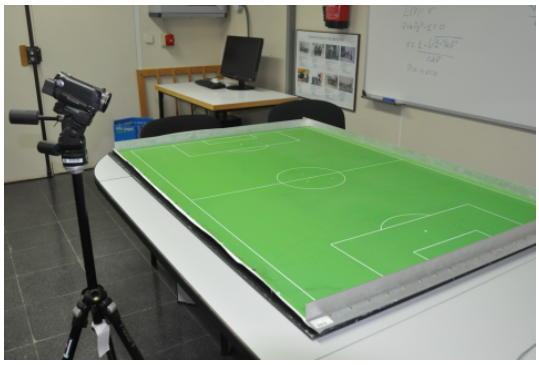

Fig. 8 Video camera mounted on a tripod and scale soccer court model.

5.2 Camera motion tracking experiments setup

In this section we explain the experiments carried out for both, the scale soccer court model sequence (823 frames) with size 1440×812 pixels, and the real sequence (385 frames) from a soccer match in HD (1920×1080 pixels). The scale model sequence was taken using a domestic video camera (Sony HDR-SR5E) mounted on a tripod as it is shown in Fig. 8. The real soccer match sequence was recorded with a professional camera, this video was provided by a TV producer and the characteristics were not given.

For each video sequence we need some information before starting the camera motion estimation, as ex-

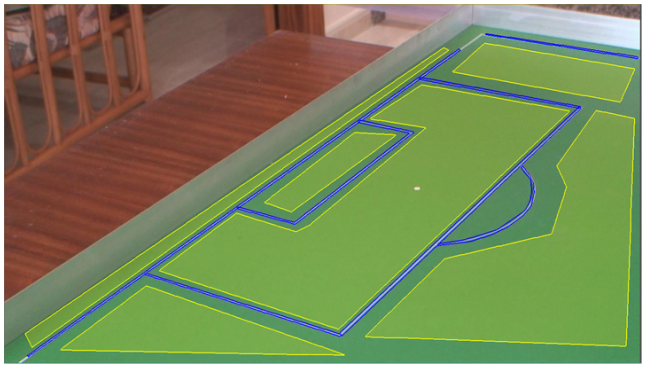

Fig. 9 Two different classes are used in the segmentation: white primitives and grass. Grass is segmented using polygons, whereas segments are used for primitives. Scale model image.

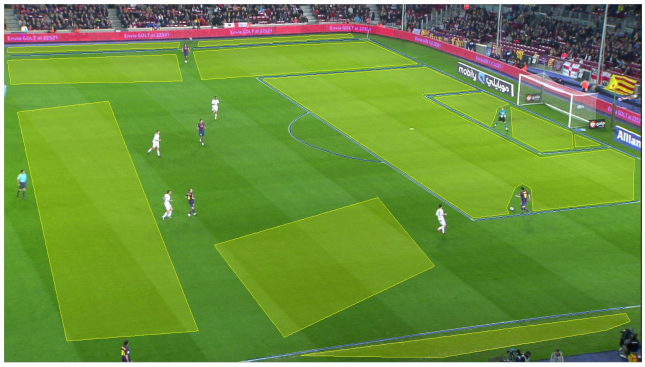

Fig. 10 Two different classes are used in the segmentation: white primitives and grass. Grass is segmented using polygons, whereas segments are used for primitives. Real image.

plained in Sects. 3 and 4.2. One of these requirements is the training data set for the decision tree. This data set is obtained by segmenting the first frame of the sequence with only two different classes: primitives (white lines and circles) and grass. Examples of segmentation are shown in Figs. 9 and 10.

The other requirement to start the tracking stage consists of calculating the tripod calibration geometry, which is estimated from some frames extracted at different instants of the video sequence. The calibration of those frames is computed with standard calibration techniques, obtaining both, intrinsic and extrinsic parameters. The selected frames to obtain the tripod geometry are shown in Fig. 11, where it can be appreciated from the projection of the primitives on the image that the camera calibration is efficiently estimated.

When the initialization stage is done, we can continue with the camera motion tracking in the whole video sequence. In Figs. 12 and 13, we can observe some tracking results from some extracted frames. See on the left, images with the tracked primitive centers, and on the right the reference field projected onto the image using the computed homography. This homography was calculated from the tripod geometry and primitives location as explained in Sect. 4.4.

5.3 Camera motion tracking results

To show the performance of the proposed method, we have tracked the camera motion in two different sequences. The processing time for these sequences are in Table 4. All experiments were executed in an Intel Core i7 2.00 GHz processor with 4 GB RAM. The camera motion tracking procedure for a 1440×812 frame lasts around 3 milliseconds for the scale model sequence and around 5 milliseconds for the real sequence (frame size is 1920×1080). For the times shown, the primitive tracking and camera parameters computation are also included but the image loading time is not considered due to its strong dependency on the system architecture. From that, it is clear that the proposal can be applied to real time video processing.

Results shown in Table 4 are for a multi-thread and sequential implementations of the primitive tracking stage. The multi-thread version is a simple parallelization that consists of assigning the calculation related to each primitive to a thread using OpenMP. Specifically only the simple OpenMP sentence *parallel for*, which assigns to each thread an iteration of the loop, was used [33]. However, the sequential implementation is also able to process the frames in real time (see Table 5).

We observe that our method is faster than the one proposed in [1], which requires 5.8 milliseconds for processing a 720×576 frame, running on a 3.4 GHz Pentium 4, but there is processed only half of the image lines (actually 720×288 frame). Moreover, we can see that our method is even faster than the recent work described in [18] which requires 4.5 ms per 720×576 frame on a standard PC and using GPU implementation of the Hough transform (under Windows XP SP3, processor Intel Core 2 Quad 2.4 GHz, 2GB RAM and 2xNVIDIA 8800 GTX as graphic board). A comparison between the processing times of the different methods is shown in Table 6.

We note that, in some video frames, there are only a few visible primitives in (see Figs. 14, 15). For example in the scale model sequence there are 51 frames with only 3 visible primitives. The histograms in Figs 14, 15 show that using our proposal, few primitives are enough for an efficient calibration. In that case, standard techniques to recover the camera motion parameters fail because they require more robust information, for example they need at least four visible lines or four points, etc. Since we deal with cameras mounted on a tripod, the restrictions due to the geometry of the tripod strongly simplify the problem and allow to use the primitive tracking to accurately recover the camera motion parameters, even for the case that only two primitives are visible.

The experimental results clearly demonstrate that the camera parameters obtained by our method are very

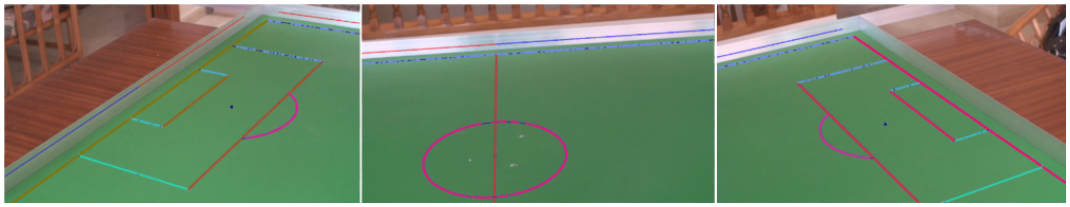

Fig. 11 Tripod geometry calibration for three reference frames. To validate the results, soccer field primitives are projected onto the real images using different colors.

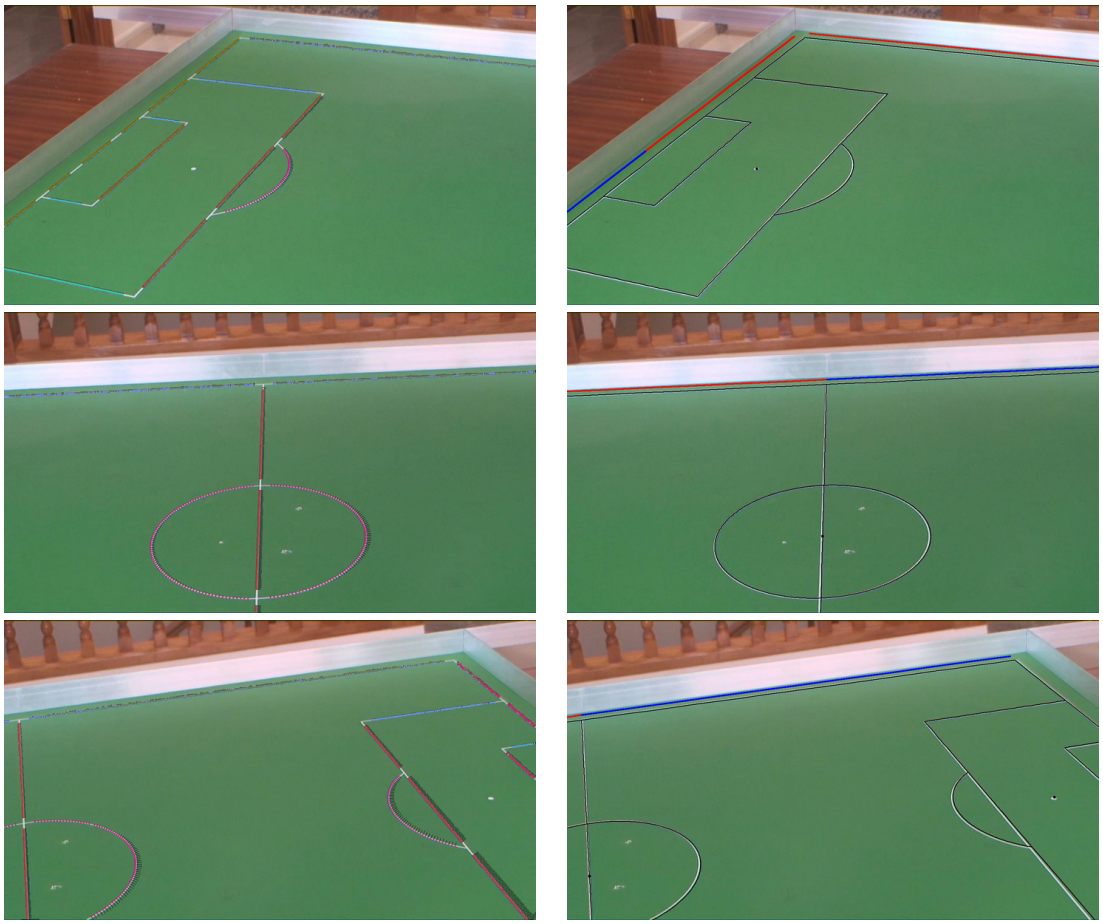

Fig. 12 Camera motion tracking results. Scale model: Images on the left show the tracked lines. Images on the right illustrate where the reference soccer field was projected onto the image using the calculated camera parameters.

accurate. This is an important issue for a subsequent visual processing task. As we can observe in Table 7, which shows the numerical error of the results, the projection errors are very small in both types of sequences. We measure the accuracy with the average squared Euclidean distance between the projected points and their corresponding soccer field primitives.

To illustrate the quality and accuracy of the camera motion tracking, we have used the camera parameters obtained in the tracking procedure to insert some computer generated graphics into the video. When the camera motion and intrinsic parameters are known, we can synchronize the real camera with a virtual camera and render objects with the same perspective. These experiments were chosen, because this kind of application requires an accurate and fast camera motion tracking computation. The results of these experiments are shown in Fig. 16. Those frames were extracted from the video sequences hosted in http://www.ctim.es/demo108/.

6 Conclusions

The main contribution of this paper is a new method to obtain real time camera calibration in planar view scenarios. The proposed method is a combination of the

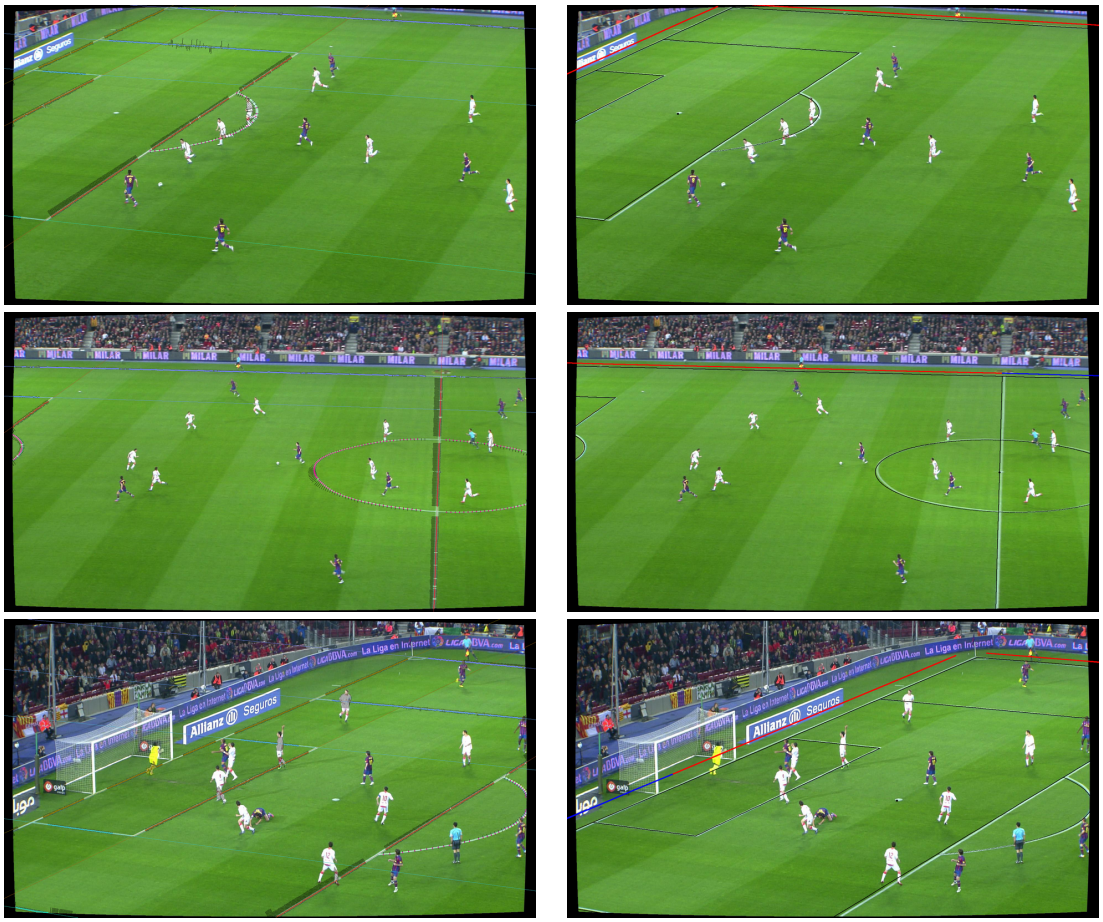

Fig. 13 Camera motion tracking results. Real images: Images on the left show the tracked lines. Images on the right illustrate where the reference soccer field was projected onto the image using the calculated camera parameters.

Table 4 Average processing time for both sequences in milliseconds per frame. The scale model sequence contains 823 frames with size 1440 × 812 pixels. The real sequence is 385 frames long and the frame size is 1920 × 1080.

Stages	**Scale model sequence**	**Real sequence**
Primitive tracking	1 ms/frame	2 ms/frame
Camera parameters computation	2 ms/frame	3 ms/frame
Camera motion tracking (total)	3 ms/frame	5 ms/frame

Table 5 Average processing time for different primitive tracking implementations, sequential and parallel in milliseconds per frame.

Primitive tracking implementation	**Scale model sequence**	**Real sequence**
Sequential	2 ms/frame	5 ms/frame
Parallel	1 ms/frame	2 ms/frame

tripod geometry restrictions to strongly simplify the camera motion, a CART method designed to extract image primitives in real time and an iterative procedure to camera motion estimation. One important added value of the proposed new method is that it can deal with complex calibration scenarios where a very few primitives are visible.

We first assumed that the camera is mounted on a tripod, which is a common situation in practice, and study the geometry of the tripod from a mathematical point of view. This assumption strongly simplifies the calibration problem and allows recovering the frame calibration in situations where general calibration techniques fail. One of the main novelties of the tripod model is the fact that the tripod rotation and camera projection centers do not necessarily coincide.

Concerning the experimental results, the most relevant conclusion is the high accurate camera calibration we obtain in real time using the proposed method. This accuracy is showed in two ways: on the one hand

Table 6 Comparison between different methods performance.

Method	Processor	frame size	ms/frame	ms/pixel
Thomas [1]	3.4 GHz Pentium 4	720×288	5.8	2.7970e-005
Battikh et al. [18]	Intel Core 2 Quad 2.4 GHz	720×576	4.5	1.0850e-005
Authors (Sequential)	Intel Core i7 2.00 GHz	1440×812	3	2.5656e-006

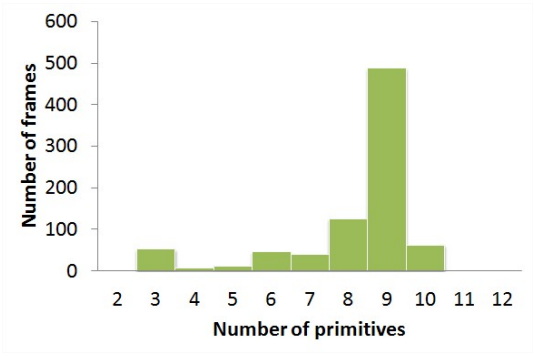

Fig. 14 Number of frames with respect to the number of primitives used to calibrate. Scale model sequence.

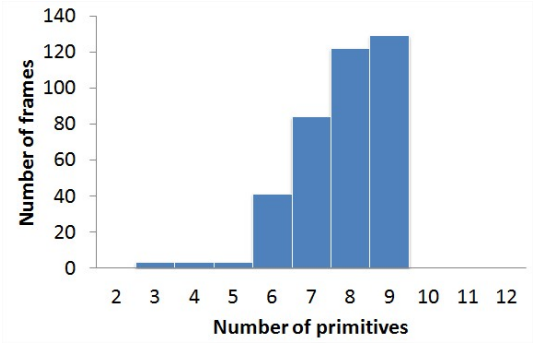

Fig. 15 Number of frames with respect to the number of primitives used to calibrate. Real match sequence.

Table 7 Average squared error (in meters, given by Equation (15)) between the projection of the extracted primitive points and the soccer field primitives using the camera motion tracking

Sequence	Error
Scale model	0.01610
Real	0.01468

the primitive reprojection error showed in Table 7 is very small which is a quantitative quality evaluation criterium; on the other hand, we show some videos where some artificial objects have been inserted in the sequences using the proposed method. The good visual quality of the position of such graphic object through the sequence is a very good qualitative evaluation of the results. These videos are shown in www.ctim.es/demo108.

We present experiments using HD videos of sport events (soccer matches) in both, scale soccer court models and real scenarios. The experimental results show that the method is real time. The average processing time is around 5 milliseconds per HD frame. In terms of computational complexity, the main novelties are that the computation of a decision tree is very fast and the method is local, i.e., we only need to process a neighborhood around the primitive location in the previous frame.

Future work would consider a new decision tree strategy to suit the possible light condition variations through the video sequence.

Acknowledgements This research work has been partially supported by the MICINN project reference MTM2010-17615 (Ministry of Science and Innovation, Spain). We acknowledge MEDIAPRODUCCION S.L. for providing us with the real HD video we use in the experiments.

References

1. G. Thomas: Real-time camera tracking using sports pitch markings. Journal of Real-Time Image Processing, 2, 2-3, (2007) 117–132.
2. J. Chandaria, G. Thomas, D. Stricker: The MATRIS project: real-time markerless camera tracking for Augmented Reality and broadcast applications. Journal of Real-Time Image Processing, 2, 2-3, (2007) 69–79.
3. P. Santos, A. Stork, A. Buaes, C. Pereira, J. Jorge: A real-time low-cost marker-based multiple camera tracking solution for virtual reality applications. Journal of Real-Time Image Processing, 5, 2, (2009) 121–128.
4. L. Agapito, E. Hayman, I. Reid: Self-calibration of rotating and zooming cameras. International Journal of Computer Vision, 45, (2001) 107–127.
5. R. Hartley: Self-calibration from multiple views with a rotating camera. European Conference on Computer Vision, 800, (1994) 471–478.
6. I. Junejo, H. Foroosh: Practical PTZ camera calibration using Givens rotations. 15th IEEE International Conference on Image Processing, (2008) 1936–1939.
7. A. Senior, A. Hampapur , M. Lu: Acquiring multi-scale images by pan-tilt-zoom control and automatic multi-camera calibration. In: Seventh IEEE Workshop on Applications of Computer Vision, (2002) 433–438.
8. J. Davis, X. Chen: Calibrating pan-tilt cameras in wide-area surveillance networks. 9th IEEE International Conference on Computer Vision, (2003) 144–149.
9. H. Li, C. Shen: An LMI approach for reliable PTZ camera self-calibration. IEEE International Conference on Video and Signal Based Surveillance, (2006).
10. A. Basu, K. Ravi: Active camera calibration using pan, tilt and roll. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 27, 3, (1997) 559–566.
11. E. Hayman, D. Murray: The effects of translational misalignment when self-calibration rotating and zooming cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25, 8, (2003) 1015–1020.
12. JB. Hayet, J. Piater: On-Line rectification of sport sequences with moving cameras. In: MICAI: Advances in Artificial Intelligence, 4827, (2007) 736–746.

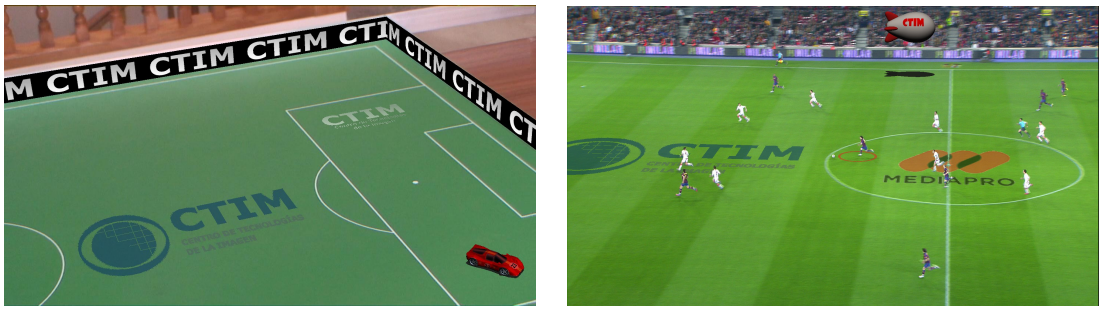

Fig. 16 Graphic insertion using the parameters calculated with real-time camera motion tracking method. Extracted from the real video provided as supplementary material (www.ctim.es/demo108).

13. H. Kim, KS. Hong: Robust image mosaicing of soccer videos using self-calibration and line tracking. In: Pattern analysis and applications, 4, (2001) 9–19.
14. Q. Li, Y. Luo: Automatic camera calibration for images of soccer match. In: Proceedings of World Academy of Science, Engineering and Technology, 1, (2005) 170–173.
15. H. Sankoh, M. Sugano, S. Naito: Dynamic camera calibration method for free-viewpoint experience in sport videos. In: MM 12 Proceedings of the 20th ACM international conference on Multimedia, (2012) 1125–1128.
16. Y. Watanabe, M. Haseyama, H. Kitajima: A soccer field tracking method with wire frame model from TV images. In: International Conference on Image Processing, 1-5, (2004) 1633–1636.
17. D. Farin, J.G. Han, PHN. de With: Fast camera calibration for the analysis of sport sequences. In: IEEE International Conference on Multimedia and Expo (ICME), 1-2, (2005) 482–485.
18. T. Battikh, I. Jabri: Camera calibration using court models for real-time augmenting soccer scenes. In: Multimedia Tools Applications, 51 (2011) 997–1011.
19. V.B. Kashany, H.R.I. Pourreza: Camera parameters estimation in soccer scenes on the basis of points at infinity. In: IET Computer Vision, 6, 2, (2012) 133–139.
20. X. Gao, Z. Niu, D. Tao, X. Li: Non-goal scene analysis for soccer video. In: Neurocomputing, 74, 4, (2011) 540–548.
21. B. Jiang, L. Songyang and B. Liang: Automatic line mark recognition and its application in camera calibration in soccer video. IEEE International Conference on Multimedia and Expo (ICME), 1–6, (2011).
22. L. Breiman, JH. Friedman, RA. Olshen, CJ. Stone: Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.
23. M. Ozuysal, M. Calonder, V. Lepetit, P. Fua: Fast Keypoint Recognition using Random Ferns. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, (2010).
24. V. Lepetit, P. Fua: Keypoint Recognition using Randomized Trees. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, 9, (2006) 1465–1479.
25. M. Emre Celebi and H. Iyatomi and W. V. Stoecker and R. H. Mossd and H. S. Rabinovitz and G. Argenziano and H. P. Soyer: Automatic detection of blue white veil and related structures in dermoscopy images. In: Computerized Medical Imaging and Graphics, 32, (2008) 670–677.
26. G. Macchiavello, G. Moser, G. Boni, S. B. Serpico: Automatic unsupervised classification of snow-covered areas by decision-tree classification and minimun error thresholding. In: IEEE International Geoscience and Remote Sensing Symposium, 1-5, (2009) 1251–1254.
27. Z.Zili, Q.Qiming, G. Junping, D. Yuzhi, Y. Yunjun, W. Zhaoqiang, D. Fanwei: CART-Based Rare Habitat Information Extraction For Landsat ETM+. In: IEEE International Geoscience and Remote Sensing Symposium, 3, (2008) 1071–1074.
28. L. Alvarez, L. Gomez, P. Henriquez, L. Mazorra: Automatic camera pose recognition in planar view scenarios. In: LNCS, 17th Iberoamerican Congress on Pattern Recognition.(CIARP), 7441, (2012) 406–413.
29. L. Alvarez, L. Gomez, J.R. Sendra: Accurate Depth dependent lens distortion models: an application to planar view scenarios. In: Journal of Mathematical Imaging and Vision, 39, (2011) 75–85.
30. D. Pena: Análisis de datos multivariantes. Madrid, 2002.
31. T. Hastie, R. Tibshirani, J. Friedman: The Elements of Statistical Learning. Canada, 2002.
32. M. Aleman-Flores, L. Alvarez, P. Henriquez, L. Mazorra: Morphological thick line center detection. In: LNCS, 7th International Conference on Image Analysis and Recognition. (ICIAR), 6111, (2010) 71–80.
33. B. Chapman, G. Jost, R. van der Pas: Using OpenMP: Portable Shared Memory Parallel Programming, 2007.