Research White Paper

WHP 168

July 2008

Real-Time Camera Tracking using Sports Pitch Markings

G. A. Thomas

BRITISH BROADCASTING CORPORATION

Real-Time Camera Tracking using Sports Pitch Markings

G. A. Thomas

Abstract

When broadcasting sports events such as football, it is useful to be able to place virtual annotations on the pitch, to indicate things such as distances between players and the goal, or whether a player is offside. This requires the camera position, orientation, and focal length to be estimated in real time, so that the graphics can be rendered to match the camera view. Whilst this can be achieved by using sensors on the camera mount and lens, they can be impractical or expensive to install, and often the broadcaster only has access to the video feed itself. This paper presents a method for computing the position, orientation and focal length of a camera in real time, using image analysis. The method uses markings on the pitch, such as arcs and lines, to compute the camera pose. A novel feature of the method is the use of multiple images to improve the accuracy of the camera position estimate. A means of automatically initialising the tracking process is also presented, which makes use of a modified form of Hough transform. The paper shows how a carefully-chosen set of algorithms can provide fast, robust and accurate tracking for this real-world application.

This paper was published in the Journal of Real Time Image Processing, Vol. 2, No. 2-3, November 2007, pp. 117-132. It is an extended version of a similar paper presented at CVMP2006, published as WHP146.

Additional key words: camera tracking, pose estimation, football, soccer

Real-Time Camera Tracking using Sports Pitch Markings

G.A. Thomas

BBC Research, Kingswood Warren, Tadworth, Surrey, UK

Abstract

When broadcasting sports events such as football, it is useful to be able to place virtual annotations on the pitch, to indicate things such as distances between players and the goal, or whether a player is offside. This requires the camera position, orientation, and focal length to be estimated in real time, so that the graphics can be rendered to match the camera view. Whilst this can be achieved by using sensors on the camera mount and lens, they can be impractical or expensive to install, and often the broadcaster only has access to the video feed itself. This paper presents a method for computing the position, orientation and focal length of a camera in real time, using image analysis. The method uses markings on the pitch, such as arcs and lines, to compute the camera pose. A novel feature of the method is the use of multiple images to improve the accuracy of the camera position estimate. A means of automatically initialising the tracking process is also presented, which makes use of a modified form of Hough transform. The paper shows how a carefully-chosen set of algorithms can provide fast, robust and accurate tracking for this real-world application.

Keywords camera tracking, camera calibration, pose estimation, football, soccer

1. Introduction

One key aim of a broadcaster covering a sports event is to help “tell a story” to the viewers, by giving insight into what is happening, and analyzing the action. To help with sports analysis, a common requirement is to be able to overlay graphics on the image, which appear to be tied to the ground. It is also useful to be able to show markings at the correct absolute scale, such as distances from a player to the goal. This requires knowledge of the camera pose (position and orientation), as well as the focal length (or zoom), i.e. a full metric camera calibration. The calibration data generally needs to be generated at full video rate (50Hz or 60Hz). Examples of some typical overlaid graphics are shown in Figure 1.

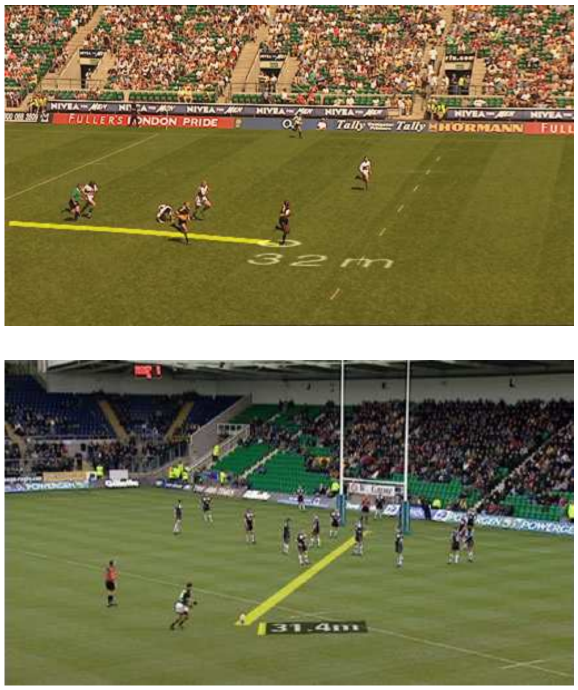


Figure 1 - Graphics overlaid using camera pose data

The camera pan, tilt and zoom need to be estimated to a sufficient accuracy so that overlaid graphics appear stable and accurate to around 1 pixel or better in the image. However, there is generally no need to make a real-time estimate of the camera position, roll, or lens distortion: Cameras at sports events are typically mounted on fixed pan/tilt heads (Figure 2). Although the principal point of the lens may move by 20-30cm as the camera pans (as the axis about which it pans is generally a little way behind the lens), this movement is usually negligible compared with the scale of the graphics being added (which may have line thicknesses equivalent to around 0.5m as shown in Figure 1). The camera mount prevents the camera from rolling (i.e. it cannot rotate about its direction-of-view). Although there will usually be a small amount of lens distortion, particularly at wide angles, most commercial real-time rendering software currently does not correct for it, so there is generally no need to estimate it for applications involving real-time graphics insertion. Whilst real-time graphics overlay is the primary application considered in this paper, there are other applications where more accurate camera calibration is needed, for example multi-camera 3D reconstruction [5]. For these, an estimation of lens distortion and principal point position are likely to be needed.

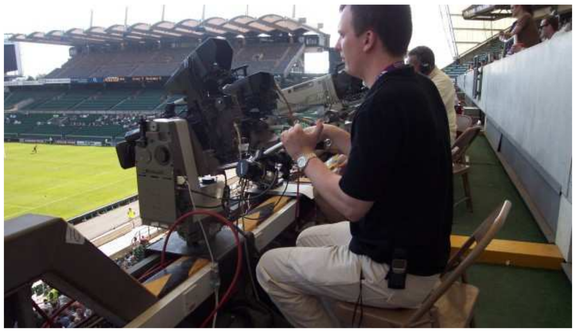


Figure 2 - A TV camera at the Twickenham Rugby Stadium, UK

One way in which camera calibration data can be derived is by performing an initial off-line calibration of the position of the camera mounting using a theodolite or range-finder, and mounting sensors on the camera and the lens to measure the pan, tilt, and zoom. However, this is costly and sometimes very difficult, for example if it is not possible to gain access to the camera mount, or if the camera is mounted on a non-rigid structure such as a crane.

A more attractive way of deriving calibration data is by analysis of the camera image. Camera calibration is typically carried out using a calibration object having easily-identifiable features in accurately-known locations, for example a planar checkerboard pattern [14]. The lines on a sports pitch are usually in known positions, and these can be used to compute the camera pose. In sports such as football, the layout of some pitch markings (such as those around the goal) is fully specified, but the overall dimensions vary between grounds. For example, the English Football Association specifies that the pitch length must be in the range 90-120m, and the width 45-90m; for international matches, less variation is allowed (length 100-110m and width 64-75m) [4]. It is thus necessary to obtain a measurement of the actual pitch.

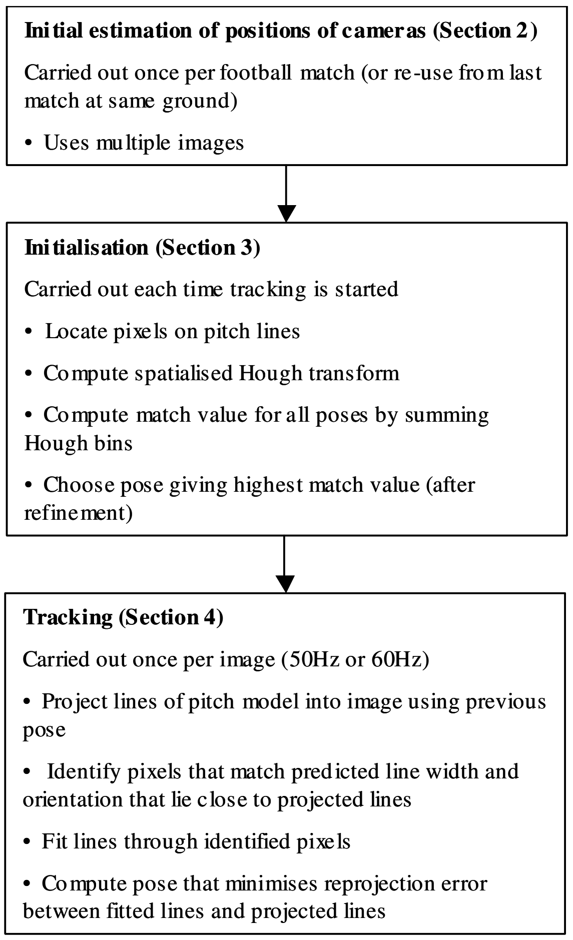
One example of past work in this area is [16], in which an exhaustive search over position, orientation and field-of-view is performed to match a wire-frame model of the pitch lines to the lines detected in the camera image. Although this method requires no manual initialisation, the exhaustive search approach suggested is likely to result in processing times that are too long to be practical for applications in sports graphics generation, where the system needs to initialise in about 1 second, and track at 50-60 frames per second. While the quantisation inherent in the searching process in [16] may be sufficient for the application described (gathering statistics on player position), it is unlikely to result in a camera pose that has sufficient accuracy for convincing graphical overlay.

An example of a method that computes the camera pose by iterative minimisation of the error between observed lines and reprojected lines from a model is [15]. This method works by locating edge points in the image close to the predicted edge position, but no method of automatic initialisation is presented. It is stated that the method tracks with a small amount of jitter, which can be reduced by using texture information for tracking in addition to the edge information.

Other researchers have investigated the detection of specific kinds of line features that occur in football. For example, [17] presents a method for detecting the ellipse in the image formed by the centre circle, although this only works when more than half of the circle is visible, and makes no use of other line features in the image. Billboards around the pitch can also be tracked [1], although many football grounds now have animated or scrolling billboards, which would make such an approach unsuitable.

In this paper we propose a method based on line tracking, similar to [15] in that the camera pose is computed to minimise the reprojection error to observed edge points. We also use a variant of the multi-hypothesis approach that [15] describes: only those edge points closest to the predicted line position are considered, rather than using all points within a given search area. This provides robustness to the appearance of other nearby edge points.

However, our method includes an automatic initialisation process, similar in concept to the exhaustive search of [16], but implemented in such a way that the process can be carried out in about one second. We also take advantage of the fact that TV cameras at outside broadcasts are often mounted on fixed pan/tilt heads, so that their position remains roughly constant over time. This paper extends the results previously reported in [13]. An overview of the whole process is given in Figure 3.



Initial estimation of positions of cameras (Section 2)

Carried out once per football match (or re-use from last match at same ground)

- Uses multiple images

Initialisation (Section 3)

Carried out each time tracking is started

- Locate pixels on pitch lines
- Compute spatialised Hough transform
- Compute match value for all poses by summing Hough bins
- Choose pose giving highest match value (after refinement)

Tracking (Section 4)

Carried out once per image (50Hz or 60Hz)

- Project lines of pitch model into image using previous pose
- Identify pixels that match predicted line width and orientation that lie close to projected lines
- Fit lines through identified pixels
- Compute pose that minimises reprojection error between fitted lines and projected lines

Figure 3 - Overview of the method

The following section describes the method used to estimate the position of each camera mounting, when the system is first set up. Section 3 explains the initialisation method, which is invoked when starting to track, or when the system has lost track (for example, if the camera view moved away from showing any pitch lines). Section 4 discusses the real-time tracking process. Section 5 presents some results, and the remaining sections present discussions and conclusions.

2. Estimation of the camera position

Most cameras covering events such as football generally remain in fixed positions during a match. Indeed, the positions often remain almost unchanged between different matches at the same ground, as the camera mounting points are often rigidly fixed to the stadium structure. A typical camera mounting was shown in Figure 2. It therefore makes sense to use this prior knowledge to compute an accurate camera position, which is then used as a constraint during the subsequent tracking process.

Estimating camera position from image correspondences can be a poorly-constrained problem, particularly if focal length also needs to be estimated when using a planar calibration object. This is because the effect on the image of a small change in focal length is similar to a translation of the camera along its direction of view. If the variation of depth in the image is small compared to the distance from the camera to the scene (for example, in the case of a planar scene whose normal is nearly parallel to the direction of view), the two effects are essentially indistinguishable. This is why some camera calibration methods such as [18] propose the use multiple images of an inclined planar calibration object in order to improve their accuracy: they constrain the internal parameters of the camera (focal length and lens distortion) to be the same in all images, but allow the external parameters (position and orientation) to vary. A single optimization process estimates the common internal parameters and the separate external parameters for all images. In our application, we need to compute the orientation and focal length of images in real time as they arrive, so it is not immediately apparent that an approach using multiple images is suitable. However, as the main cause of uncertainty comes when trying to estimate both the focal length and position simultaneously from a single image, we can get the benefits of a multiple-image approach by pre-computing the camera position using multiple images grabbed with a wide range of pan and tilt angles, and constraining the position to this estimated value when computing pan, tilt and focal length using an individual image during real-time tracking.

The pose computation method described in Section 4 is used to compute the camera position, orientation and field-of-view, for a number of different camera orientations, covering a wide range of pan angles. The pose for all images is computed in a single optimisation process, constraining the position to a common value for all the images, but estimating separate values of pan, tilt and focal length for every image. This significantly reduces the inherent ambiguity between the distance of the camera from the reference features and the focal length, even though the focal length is allowed to vary in each image. By including views of features in a wide range of positions (e.g. views of both goal areas), the uncertainty lies along different directions, and solving for a common position allows this uncertainty to be significantly reduced. Examples of the position estimation process are given in Section 5.1.

Mounted cameras usually cannot roll (i.e. rotate about their direction of view). However, this does not necessarily mean that the roll angle can be assumed to be zero. The camera may not necessarily be mounted flat on the pan/tilt head, or the head itself may not be aligned with the pan axis exactly vertical. Also, it is usually assumed that the plane of the pitch is horizontal, but this may not be the case. Each of these effects can give rise to what appears to be a small amount of camera roll, which could vary with pan or tilt, depending on the cause. One solution would be to compute camera roll for every image during the tracking process, but this introduces an additional degree of freedom, and therefore will increase the noise sensitivity and increase the minimum number of features needed for accurate pose computation. Another option would be to attempt to solve for each of these small mis-alignments separately. Instead, we chose to solve for the apparent rotation of the pitch plane about the dominant direction of the camera view (for example, allowing the pitch to rotate about the z axis for a camera whose direction-of-view when looking at the centre of the pitch is closely parallel to the z axis). We found that in practice this accounts sufficiently well for the combination of effects described earlier. The pitch rotation is computed during the global position computation process, giving a single value optimised for all the images used.

Although it is generally best to apply this technique by manually selecting around 10-20 images from each camera (for example, by picking images from a video tape recorded from a previous game at the ground, that show a large number of lines and cover a wide range of pan and tilt values), we have developed a method whereby images can be acquired automatically to refine the position computation during the tracking process. The camera position and orientation are first set manually to give good alignment between actual and projected pitch lines for a given image. The tracking process (Section 4) then computes new values of pan, tilt and roll for every image, using the previous pose as an initial estimate. Each time a pose is computed that has values of pan or tilt that are significantly different from those of images used previously (typically by about 10º), the image is added to the list of images used for global position computation, and a new globally-consistent position and pitch plane rotation are calculated, incorporating the newly-captured image. This allows the system to automatically refine its estimated position during tracking.

The camera position and pitch rotation computed in this way are then used for the initialisation and tracking processes described below.

3. Initialisation

3.1 Approach

The Hough transform [7] is a well-known way of finding lines in an image. It maps a line in the image to a point (or accumulator "bin") in Hough space, where the two axes represent the angle of the line and the shortest distance to the centre of the image. If the camera pose is known roughly, it is possible to predict which peak in Hough space corresponds to which known line in the world, and hence to calibrate the camera. However, if the camera pose is unknown, the correspondence can be difficult to establish, as there may be many possible permutations of correspondences. Furthermore, if some lines are curved rather than straight, they will not give rise to a well-defined peak and are thus hard to identify.

Rather than attempting to establish the correspondence between world lines and peaks in Hough space, we use the Hough transform as a means to allow us to quickly establish a measure of how well the image matches the set of lines that would be expected to be visible from a given pose. A "match value" for a set of lines can be obtained by adding together the set of bins in Hough space that correspond to the lines we are looking for. Thus, to test for the presence of a set of N lines, we only have to add together N values from the Hough transform, rather than examining all the pixels in the image that we would expect the lines to lie upon.

We use this in an exhaustive search process, to establish the match value for each pose that we consider. This provides a much faster way of measuring the degree of match than that used in the exhaustive search process proposed in [16]. For each pre-determined camera position, we search over the full range of plausible values of pan, tilt, and field-of-view, calculating the match value by summing the values in the bins in the Hough transform that correspond to the line positions that would be expected.

By representing a curved line as a series of line segments, curves can also contribute to the match, even if they do not give rise to local maxima in the Hough transform. We used one segment for every 20º of arc. Although specific forms of Hough transform exist for circle or ellipse detection (such as that used in [17]), we chose the line segment approach to allow both curves and lines to be handled in a single process.

The following two sub-sections explain the kind of Hough transform used, and give further implementation details.

3.2 A variant on the Hough transform that maintains spatial information

A point in a conventional Hough transform represents a line of infinite length, i.e. the information about which part of the line a contributing point lies on is lost. This means that, when measuring the degree of match for a line segment from the value in the corresponding Hough bin, all edge pixels that are co-linear with this segment will be considered, even if they lie beyond the ends of the line segment. This makes the match value less reliable, as noise samples, or samples from other lines, will contribute when they should not. In particular, the peak in the Hough transform from a short line segment (or a short segment of a curve) may be no higher than a peak caused by samples from several other line segments and from the limbs of a player, that coincidentally happen to be co-linear. That is, the number of pixels giving a significant output from the line detector that lie along a short genuine line segment may be no larger than the number that lie on a longer 'imaginary' line that passes through players and parts of other lines. The presence of a short line segment could thus be incorrectly inferred in this situation. The information that the genuine peak came from samples in a specific localized area, whilst the other came from a spatially diverse area, is lost in a conventional Hough transform.

Various methods of incorporating spatial information in a Hough transform have been proposed before; for example, [6] describes an approach in which the input image is divided into a quad-tree, so that the lines in a particular region can be identified.

We chose a relatively simple approach to retaining spatial information, which we refer to here as a spatialised Hough transform. Rather than sub-dividing the image into 2D regions, we divide each line into S 1D segments: this maintains a common set of bins for the whole image, with each bin being subdivided into S sections. Rather than divide every line into S equal sections, for simplicity we divide the line by reference to either the horizontal portion of the image in which it lies (for lines that are closer to horizontal than vertical), or the vertical portion (for lines that are closer to vertical). Thus to determine the sub-bin that a given pixel contributes to, it is only necessary to examine either its x or y coordinate, depending on the angle that the bin in the transform corresponds to. The resulting transform has three dimensions: distance and angle (as in a conventional Hough transform), and "distance from picture edge", measured from either the bottom or the left of the image, depending on the slope of the line. This third axis has length S, and is in units of picture width divided by S, or picture height divided by S.

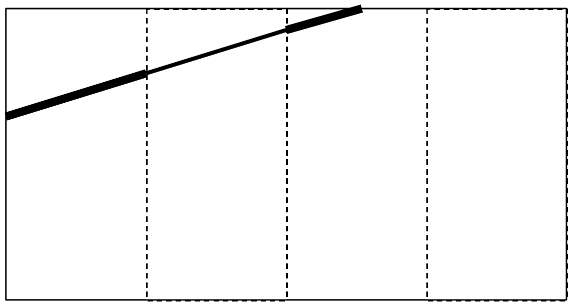


Figure 4 - Dividing a line into multiple segments for the spatialised Hough transform

Not all lines will have positions or angles that need all the bins, as they may not cover the full width or height of the image. However, this approach leads to a very efficient way of computing the sub-bin for any given pixel, and this was considered preferable to a more complicated method that might make slightly more efficient use of memory. Figure 4 shows an image in the case where S=4, and a line closer to horizontal than vertical which is thus divided horizontally. The line is divided into the three segments shown by alternating thick and thin lines; the 4th spatial sub-division of this line will never contain any points, as it would lie above the top of the image.

An example of this method being applied to a real image is shown in Figure 5. The Hough transform (a) shown on the left is a conventional transform that takes no account of the spatial extent of lines. The other two transforms in Figure 5 are from the spatial subdivisions corresponding to line segments in the upper or left part of the image (b), and the lower or right part (c). The transform in (a) is the sum of the transforms (b) and (c). The correspondences between some of the pitch lines and peaks in the Hough transform are shown. Note, for example, that the peaks for lines 2 and 4 have contributions from both spatial subdivisions (as these are long lines covering more than half the image), whereas the peak for line 3 is only visible in the bottom/right spatial subdivision. Conversely, the peak for line 1 is only visible in the top/left subdivision (b), and is easier to spot in this subdivision than in the whole transform (a), as there are less "stray" contributions from other pixels. This illustrates that by examining only the set of spatial subdivisions expected to contribute to a line, a more noise-free measurement is obtained.

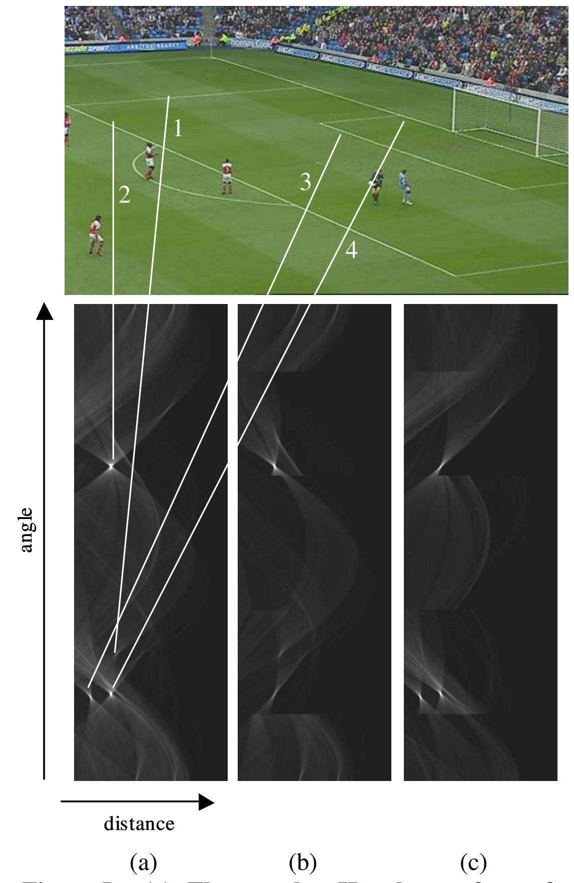


Figure 5 – (a): The complete Hough transform of the pitch lines in the image shown above; (b): just those spatial bins for the top or left half; (c): just those for the bottom or right half.

3.3 Implementation

The steps in our initialisation process are as follows:

Step 1. We first identify all the pixels in the input image that may correspond to the lines we are interested in. The line detection filter used in the tracking process described in Section 4.2 is used. However, this filter assumes the width and orientation of the line is known, which is not the case at this stage. Therefore we apply the filter several times, with a range of assumed line widths, and using both horizontal and vertical orientations. The outputs of these filtering operations are added together. For each pixel where this sum is above a given threshold, we add a value to the appropriate sub-bins of the spatialised Hough transform. The value added is proportional to the summed filter outputs, so that a higher weight is given to more reliable information.

Step 2. For each camera position estimated as described in Section 2, we step through the plausible ranges of pan, tilt and zoom, and project the lines of the pitch model into the image. The step size is chosen to be equivalent to a fixed number of pixels in the image; in the case of zoom this is the motion caused at the edge of the image. For each candidate pose where at least three lines are visible, we compute the list of sub-bins in the spatialised Hough transform that correspond to the projected lines. The contents of all the sub-bins are summed to obtain a weighting value for this pose. We make a list of the 5-10 poses having the highest sums (ignoring poses that are close to other poses having a higher sum, as they are not useful local maxima). It is worth noting that the list of sub-bins to be used for each pose depends only on the estimated camera positions and the geometry of the pitch model, so the entire list of bins for all possible poses can be pre-computed once the camera positions have been determined as described in Section 2. This significantly speeds up the search process, as no projection of pitch lines or mapping of lines to Hough space needs to be carried out during the initialisation process.

Step 3. Each of the high-scoring poses identified in the previous step are used in turn to initialise the tracking process (Section 4). Several iterations of the tracking process are run for each pose, and the pose that returned the highest match value at the end of this process is used as the final pose. The tracking process will "lock on" to lines that are within the search window it uses, and therefore this will generate a more accurate measurement of the match value for each pose than the value obtained by summing the bins in Hough space in Step 2. The search window size used in the tracking process is chosen to be slightly larger than the step size used in the exhaustive search in Step 2, to ensure that the tracking process finds the lines.

4 Tracking

4.1 Approach

The tracking process uses the pose estimate from the previous image, and searches a window of the image centred on each predicted line position for points likely to correspond to pitch lines. A straight line is fitted through each set of points, and an iterative minimisation process is used to minimise the distance in the image between the ends of the observed line and the corresponding line in the model. These steps are explained in more detail below, using the example image shown in the upper part of Figure 5.

4.2 Implementation

The steps in the tracking process are as follows:

Step 1. Each line in the pitch model is projected into the camera image using the previous camera pose as an estimate of the current pose. Figure 6 shows an example of the lines of the pitch model overlaid on the example image shown in Figure 5. Note that in this example, the actual pitch dimensions differed slightly from those assumed in the model (the actual pitch was slightly wider); this has the useful side-effect of making it easier to distinguish the actual and overlaid pitch lines in the image.

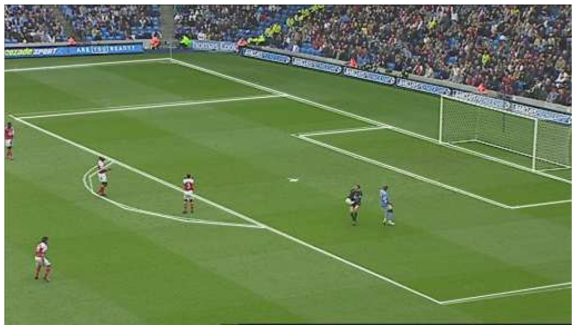


Figure 6 - Predicted line positions (solid white lines) overlaid on example image

For lines that are at least partly visible, a window around each projected line is processed to compute a measure of the extent to which each pixel is likely to lie at the centre of a line, using a simple line detection filter that uses knowledge of the predicted line width and orientation. Figure 7 shows these areas for the example image. The width of these areas is chosen to be wide enough to account for errors in the prediction of the line positions; such errors come predominantly from camera movement from the preceding frame, but also from errors in the dimensions of the pitch model. The lengths of the areas are deliberately set to be slightly shorter than the predicted line, to reduce the chance of pixels in the line it joins from being included.

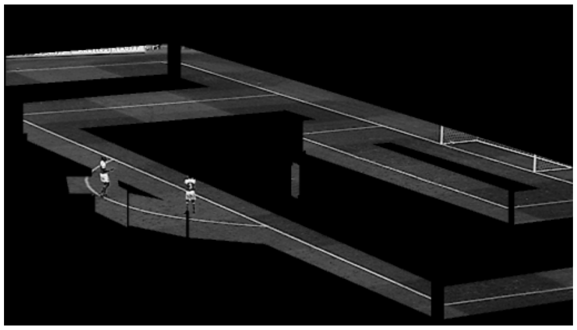


Figure 7 - Areas of the image to be searched for lines

The line detection filter operates by computing the difference between a pixel and two adjacent pixels, separated by half the width of the line we are looking for. By taking the minimum of these two difference values, the filter detects lines and ignores edges of regions significantly wider than a line. Ideally we would orient the filter so that it operated at right angles to the line we wanted to detect; however, to simplify the processing we apply the filter either horizontally or vertically, depending on whether the line is closer to horizontal or vertical. The assumed width of the line is adjusted to reflect its width in the horizontal or vertical direction, as appropriate. The local maximum of the filter output is taken to identify a pixel at the centre of the line.

The filter is applied to the blue component of the image (shown in the example in Figure 7) as this distinguishes well between the green grass and the white line; alternative methods of deriving a single-component signal from a colour image may be appropriate in applications with other line or background colours.

The adjacent pixels must also have a colour in the range expected for grass (this is to discard points in areas of the image such as the crowd or advertising hoardings). A hue-based chroma-keyer [3] is used for this purpose, which computes a key signal as a function of the difference in hue between the image pixel and a hue value set manually. The key signal is given by

$$K = X - a\,|Z| \tag{1}$$

where a is a constant which determines the acceptance angle of the keyer (i.e. its sensitivity to changes in hue), and X and Z are the colour difference signals Cb and Cr rotated through the chosen hue angle θ:

$$X = Cb \cos\theta + Cr \sin\theta \tag{2}$$
$$Z = Cr \cos\theta - Cb \sin\theta \tag{3}$$

The acceptance angle of the keyer, θ_a, is given by

$$\theta_a = 2 \tan^{-1}(1/a) \tag{4}$$

and can be set to a relatively large value in order to allow a wide variation in grass colour. The hue angle θ is chosen to correspond to a typical grass colour; due to the use of a wide acceptance angle, it does not need to be set accurately for a given situation. The output of the keyer for the image of Figure 6 is shown in Figure 8, using an acceptance angle $\theta_a = 150°$. Note that the keyer does not have to distinguish the pitch lines from the grass; nor does it have to completely key out areas such as the crowd: is function is merely to indicate areas unlikely to be grass, so that immediately adjacent areas are not considered as possible lines.

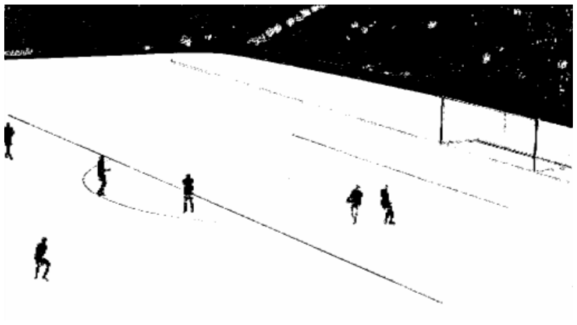


Figure 8 - Chroma keyer output

This line detection filter has the advantage of being very fast, having a simple filter kernel involving only two subtractions and a MIN operator. Figure 9 summarises its operation, and Figure 10 shows its output using the example image from Figure 6.

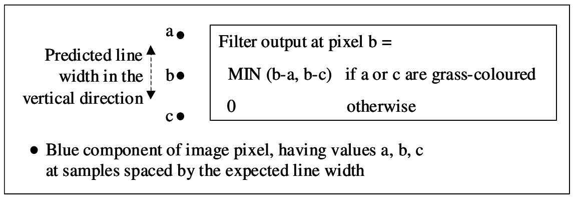

a●
Predicted line width in the vertical direction
b●
c●
Filter output at pixel b =
MIN (b-a, b-c) if a or c are grass-coloured
0 otherwise
● Blue component of image pixel, having values a, b, c at samples spaced by the expected line width

Figure 9 – The line detection filter, illustrated for the case of a near-horizontal line

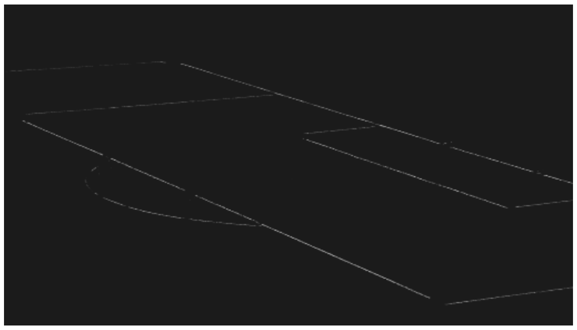


Figure 10 - Output of the line detection filter

For each column or row of pixels that the projected line crosses (depending on whether the line is closer to horizontal or vertical), we find the pixel closest to the projected line for which the line detector output is above a given threshold. The selection of the pixel *closest* to the predicted line provides robustness to the appearance of other nearby edge points (such as those of the goal post, that lie close to the goal line but further away from the predicted line position), similar to the multi-hypothesis approach in [15]. The choice of threshold value is not critical; a value of 20 grey levels (for an 8-bit image) typically allows grass texture to be discarded whilst allowing even relatively weak lines to be used.

Step 2. For each line, a straight line is fitted through the set of detected pixels, weighting each point in accordance with the amplitude of the line detector output. An overall weight is assigned to each line, based on the number of points added. The total of these weights gives a match value for this pose that can be used to trigger re-initialisation if it falls below a given threshold; it is also used when this processing is applied to pose selection in Step 3 of the initialisation process (Section 3.3). Figure 11 shows the lines fitted to the output of the line detection filter shown in Figure 10. A further example, showing how the centre circle is treated as a set of straight lines, is shown in Figure 12.

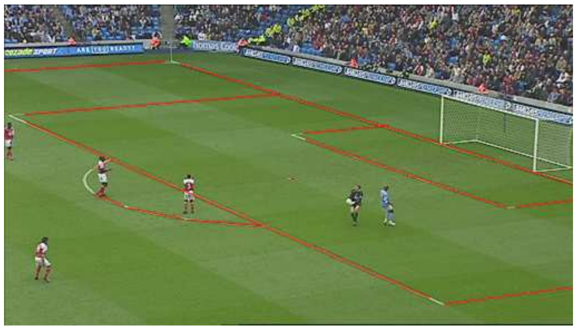


Figure 11 - Lines fitted to the detected pixels, shown in red.

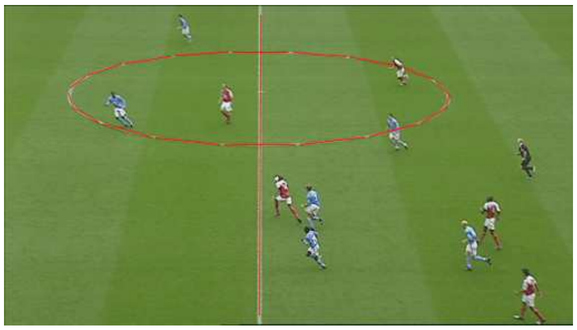


Figure 12 - Lines fitted to arcs making up the centre circle

Step 3. A least squares iterative optimisation is performed, to compute the pan, tilt and focal length (zoom) values that minimise the sum of the squares of the distance (in pixels) from the end points of the fitted lines to the lines in the pitch model projected into the camera image. For the purpose of computing the distance from an observed end-point to the projected pitch model line, the lines in the pitch model are assumed to be of infinite extent; this allows the distance to be computed in a straightforward manner. The minimisation process is similar to the well-known Levenberg-Marquardt method, using the pose from the previous image as the initial estimate, and is described in more detail below.

Rather that working with angles represented as pan, tilt and roll, we use the more conventional rotation angle definitions, as used for example in OpenGL, where a rotation is represented by a rotation θ_x (in radians) anticlockwise about the world x axis, followed by a rotation θ_y about the world y axis, and a rotation θ_z about the world z axis. Note that if we define the camera roll to be zero, and the camera pan axis to be parallel to the y axis, then $\theta_z = 0$. Let f be the focal length of the camera (in metres). We have estimates of θ_x, θ_y and f from the previous frame. We also know the position of the camera with respect to the pitch, as described in Section 2.

Let the observed image coordinates at the ends of a line fitted to a set of observed points (as derived in Step 2) be (x_{oi}, y_{oi}), where $i=\{0, 1\}$. These are expressed in units of metres with respect to the point on the camera sensor directly underneath the lens centre, after correcting for any lens distortion, using the following equations:

$$x_{oi} = (1 + d_1 r^2 + d_2 r^4)(x_p d_x - c_x) \qquad (5)$$

$$y_{oi} = (1 + d_1 r^2 + d_2 r^4)(y_p d_y - c_y) \qquad (6)$$

where

$$r^2 = (x_p d_x - c_x)^2 + (y_p d_y - c_y)^2 \qquad (7)$$

is the distance (in metres) of the observed point in the distorted image from the image centre (c_x, c_y), and (d_x, d_y) are the dimensions of a pixel. Note that for the real-time graphics overlay application discussed here, we usually assume no image distortion, and define the centre of the image as being at the middle of the captured image; however, estimation of distortion and image centre can be important in other applications, discussed later (Section 5.5).

Let the image coordinates (in metres) for the ends of the line in the model projected into the camera image using the predicted camera parameters be (x_{p1},y_{p1}) and (x_{p2},y_{p2}). These are given by the projection equations

$$x_p = -f \frac{[\mathbf{R}^T\mathbf{P}]_x}{[\mathbf{R}^T\mathbf{P}]_z} \tag{8}$$

$$y_p = -f \frac{[\mathbf{R}^T\mathbf{P}]_y}{[\mathbf{R}^T\mathbf{P}]_z} \tag{9}$$

where **P** is the position of the world point at one end of the line with respect to the camera, **R** is the rotation matrix of the camera, and $[\ldots]_x$ indicates the x component of the vector. When all angles are zero, the camera orientation is defined as looking down the –ve z axis, with y vertical.

We will write the equation of the predicted line in the image as

$$y = a + bx \tag{10}$$

where

$$a = y_{p1} - bx_{p1} \tag{11}$$

and

$$b = \frac{y_{p2} - y_{p1}}{x_{p2} - x_{p1}} \tag{12}$$

The distance of the observed line end-point i from this line (using the standard formula for the distance of a point from a line) is

$$d_i = \frac{y_{oi} - (a + bx_{oi})}{\sqrt{1+b^2}} \tag{13}$$

We make a first-order Taylor expansion of d_i', the value of d_i after making small changes $\Delta\theta_x$, $\Delta\theta_y$, Δf to the parameters we wish to adjust:

$$d_i' \approx d_i + \frac{\partial d_i}{\partial \theta_x}\Delta\theta_x + \frac{\partial d_i}{\partial \theta_y}\Delta\theta_y + \frac{\partial d_i}{\partial f}\Delta f \tag{14}$$

The derivatives of d_i may either be calculated numerically or algebraically from the projection equations. For neatness and speed, in our implementation we computed them algebraically. For example, to compute the partial derivative of d_i with respect to the camera angles, we first write (dropping the subscript i for convenience):

$$d = k(y_o - (a + bx_o)) \tag{15}$$

where

$$k = \frac{1}{\sqrt{1+b^2}} \tag{16}$$

and write

$$\frac{\partial d}{\partial \theta_x} = \frac{\partial k}{\partial \theta_x}(y_o - (a + bx_o)) - k(\frac{\partial a}{\partial \theta_x} + \frac{\partial b}{\partial \theta_x}x_0) \tag{17}$$

(noting that x_0 and y_0 do not depend on θ_x as these are the observed image coordinates, not the projected ones). We then evaluate the derivatives of a, b, and k with respect to θ_x, working forwards from the projection equations (8) and (9).

We want to minimize

$$e = \sum_{all\ lines} (\sum_{i=0,1} (d_i')^2) \tag{18}$$

summing over both endpoints of all lines, by making small changes to the camera angles and focal length. To do this, we write the equations

$$d_i' = 0 \;\Rightarrow \frac{\partial d_i}{\partial \theta_x}\Delta\theta_x + \frac{\partial d_i}{\partial \theta_y}\Delta\theta_y + \frac{\partial d_i}{\partial f}\Delta f = -d_i \tag{19}$$

as a matrix product:

$$\mathbf{A\ x = B} \tag{20}$$

where

$$\mathbf{A} = \begin{bmatrix} \frac{\partial d_0}{\partial \theta_x} & \frac{\partial d_0}{\partial \theta_y} & \frac{\partial d_0}{\partial f} \\ \frac{\partial d_1}{\partial \theta_x} & \frac{\partial d_1}{\partial \theta_y} & \frac{\partial d_1}{\partial f} \\ \frac{\partial d_2}{\partial \theta_x} & \frac{\partial d_2}{\partial \theta_y} & \frac{\partial d_2}{\partial f} \\ .. & .. & .. \\ .. & .. & .. \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} \Delta\theta_x \\ \Delta\theta_y \\ \Delta f \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -d_0 \\ -d_1 \\ -d_2 \\ .. \\ .. \end{bmatrix} \tag{21}$$

The matrix **A** contains two rows for each line that has been observed. In the notation above, the line index *i* increases as we move from one line to the next so that, for example, $i = 2$ indicates the first point on the second line. We then use a standard method for computing the least-squares solution to a set of over-determined simultaneous equations by multiplying each side of this equation by $\mathbf{A}^\mathbf{T}$:

$$\mathbf{A}^\mathbf{T}\mathbf{A}\ \mathbf{x} = \mathbf{A}^\mathbf{T}\ \mathbf{B} \tag{22}$$

$$\mathbf{x} = (\mathbf{A}^\mathbf{T}\mathbf{A}\ )^{-1}\ \mathbf{A}^\mathbf{T}\ \mathbf{B} \tag{23}$$

The calculated updates $\Delta\theta_x$, $\Delta\theta_y$, Δf are then added to the corresponding angles and focal length, and the error e is recomputed using (18). If the error has changed by less than a given threshold (corresponding to 0.01 pixel, for example), we stop iterating. It is possible that the error may have *increased* from its value at the previous iteration, if the error function is changing in a highly non-linear way and the first-order Taylor expansion becomes a poor approximation. In this case, the values of θ_x, θ_y, f are restored to their previous values, and all the updates are halved, before being reapplied. If the error e still increases, the updates are halved again, and so on until either the error decreases, or changes by less than the threshold. In practice, the error only increases if the initial estimate for the solution is a long way from the true minimum, which is not usually the case as we have a good estimate from the previous frame.

Note that when this process is being applied in the initial position estimation stage, the minimisation process also solves for the camera position and pitch orientation, and is applied simultaneously across a number of images (see Section 2). The Taylor expansion in (14) is extended to include derivatives of these extra parameters, and extra rows are added to matrix **A** from equations derived using observed and projected lines from the additional frames used for the position computation. Note that although a common position is assumed for all the cameras (so there are only 3 position unknowns to solve for, regardless of the number of frames considered), the orientation and focal length of the camera may be different in every frame, so each additional frame introduces 3 more unknowns.

As an alternative to the line-fitting process in Step 2, the set of edge points from Step 1 can be used directly in Step 3, instead of using just the two end-points of the fitted line. This increases the amount of computation in the minimisation process, as the number of equations to be minimised goes up significantly (from around 16 to about 800 in a typical case of 8 visible lines, each about 100 pixels long). The matrix **A** then contains one row for every visible point, rather than two rows for each visible line.

An advantage of using the detected line points directly is that lens distortion can be estimated. This cannot be done accurately if a straight line is fitted to each line, as information on the curvature of the line is lost. By using the edge points directly, the minimisation process can "see" the way in which adjusting the distortion parameter(s) affects the reprojection error for each point on the line, and thus the distortion can be estimated. Additional unknowns representing the distortion parameters and image centre are introduced into the minimization process, and equations (5)-(7) are used when calculating the derivatives relating to changes in these parameters. If multiple images are being used to compute a more accurate camera position, separate distortion parameters are used for each image, as the distortion is likely to change with focal length. We have found the estimation of lens distortion to be useful in applications where a very accurate calibration is needed (see Section 5.5), but for graphics overlay applications this is usually not necessary.

5 Results

The results presented in this section were obtained using broadcast video from a football match (Manchester City vs. Arsenal, 25th April 2005). Most of the material was shot from the main camera (positioned roughly in line with the centre line), with other shots from two cameras positioned roughly in line with the two 18-yard lines. Other cameras were occasionally used (for example, positioned behind the goals, or providing close-up shots of players). The material was standard broadcast quality (720x576, 50Hz interlaced, 16:9 aspect ratio, in 4:2:2 YUV format). The processing was applied to individual fields after horizontal subsampling by a factor of two, so each processed image had dimensions 360 x 288. We found that subsampling the image horizontally had a negligible effect on the accuracy of the algorithm, but reduced the processing time by around 35%. An example image from the material was shown in the top part of Figure 5.

The coordinate frame used had the origin at the centre of the pitch. The *x* axis was towards the right goal, the *y* axis pointed up, and the *z* axis pointed towards the main camera.

5.1 Computation of initial position

To illustrate the benefit of computing the camera position using multiple images, a 20-minute section of the match coverage were used. Images were manually selected, to ensure that they came from the main camera, and that they covered a wide range of pan, tilt and zoom values, in accordance with the procedure explained in Section 2.

Figure 13 shows a plot of the positions in the *xz* plane computed when each image was processed individually. The predominance of values along two diagonal lines is due to the fact that the majority of the calibration information comes from lines around the two goals, so ambiguity between changes in focal length and distance to the features used for calibration tends to be in a direction from the camera to the two goals. This can be seen more easily in Figure 14, which shows a wider view of the data points in Figure 13, including the pitch markings.

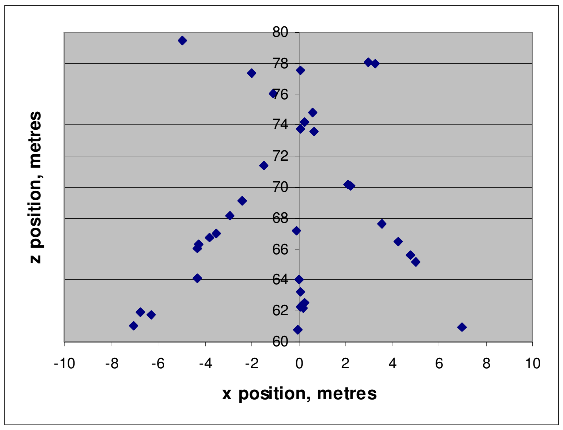


Figure 13 - Position computed using individual images

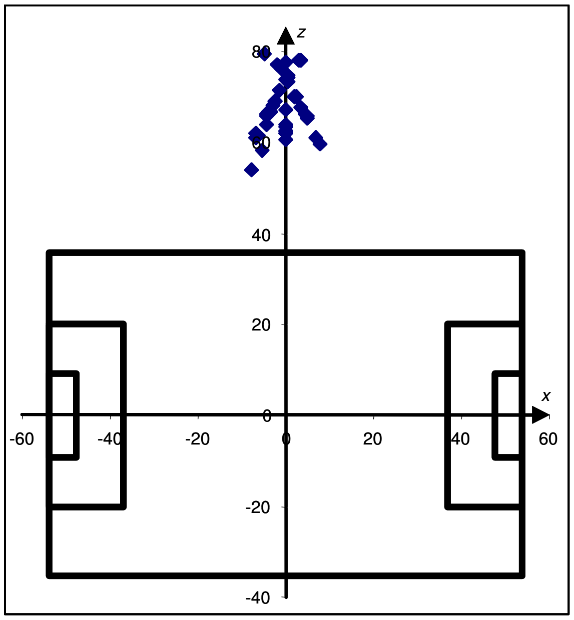
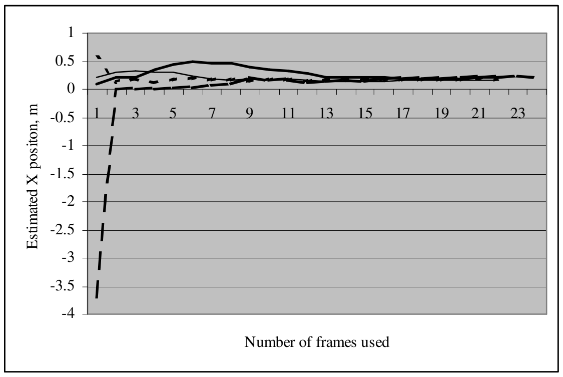


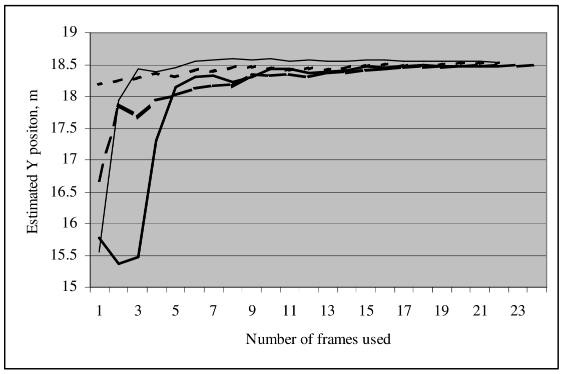
Figure 14 - Estimated camera positions in relation to the whole pitch

The position computation process was then repeated, using data from multiple images simultaneously. Each time an image was selected, the camera pose was recomputed, using both the lines visible in the new image, and all the images captured previously, forcing a common position to be used for all images. The 20-minute section of the match coverage was split into four 5-minute sections, and each was processed separately, to provide an indication of repeatability. The plots in Figure 15 show how the calculated position changed as more images were used.

It can be seen that after about 10-20 images, the computed positions stabilise to within about 0.3m of each other, to a value around (0.2, 18.5, 75.0). The largest uncertainty is in the *z* direction, which follows from the fact that the camera is on average viewing the scene within 20-30º of the *z* axis. An uncertainty in position of around +/-15cm, when the camera is about 90m from the goals, corresponds to an error of the order of 0.2%, and is of the same order as the thickness of the lines on the pitch. It is also of a similar magnitude to the likely true motion of the principal point of the camera lens, due to this point not lying on the axis about which the camera pans (see the photo of a typical camera mount in Figure 2).

It is interesting to compare the estimated position computed in this globally-consistent manner, to the positions in Figure 13 calculated using the individual images. It is worth noting that the globally-optimum position is very close to where lines through the groups of measurements would cross.





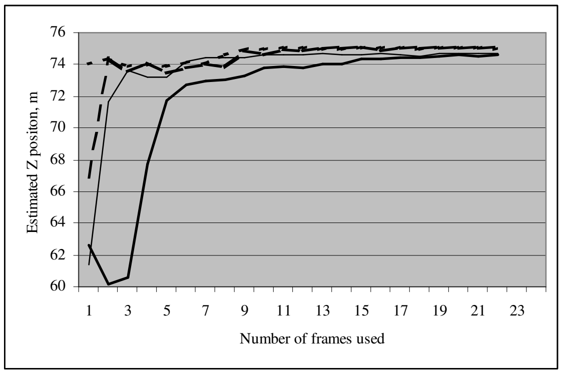


Figure 15 - Refinement of the camera position estimate (x, y and z) by using multiple images, for four sets of images

To test the validity of the assumption that a common position can be used for a wide range of viewing directions, the reprojection error (expressed as an RMS error in pixels between the ends of the measured pitch lines and the modelled lines) was measured for each new image, after using it together with the previous images for position computation, and plotted in Figure 16. The error tends to vary in an apparently random way between images; this is likely to be due primarily to inaccuracies in the position of the lines on the pitch (see for example the mismatches between projected and actual lines in Figure 6). This will vary as different lines (and even different parts of a line) appear in the shot. Without knowledge of the true position and shape of each line, it is difficult to attribute these errors to particular causes, particularly as lens distortion will also give rise to errors that vary according to how near to the edge of the image a line appears, and that depend on the degree of lens distortion present (which itself varies with both camera zoom and focus settings). The key point is that there is no significant increase in error as additional images are processed, confirming that forcing the position to be consistent with all the captured images is not over-constraining the computation.

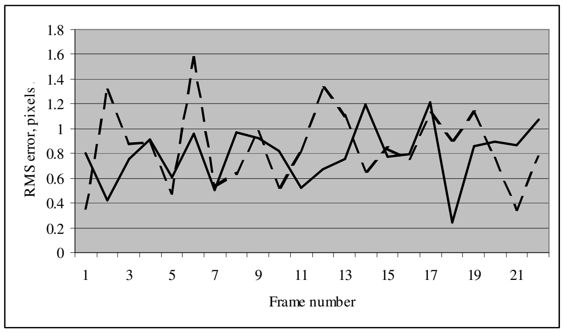


Figure 16 - RMS error for two sets of images used when progressively refining the position estimate

5.2 Initialisation

This section presents results on two aspects of the initialisation processing: the overall performance in terms of the proportion of images from which a successful initialisation can be carried out, and the benefit of retaining spatial information in the Hough transform.

The initialisation process has been used successfully on many hours of both football and rugby coverage, but for the purpose of gathering some statistics on its effectiveness, the first 35 minutes of the test material were studied in detail.

The initialisation process was configured to work with the main camera and the two 18-yard cameras, with the search parameters listed in Table 1. The step size of all search parameters was chosen to be equivalent to a movement of about 10 pixels; in the case of field-of-view, this was the figure at the edge of the image. With this set of parameters, approximately 700,000 different camera poses were searched for each camera position.

camera	*location*	*pan range*	*tilt range*	*field-of-view range*
main	(0.2, 18.5, 75.0)	-65° to 65°	-30° to -5°	4.8° to 40°
left 18-yard	(-34.7 31.7, 90.4)	-80° to 20°	-30° to -5°	4.8° to 40°
right 18-yard	(34.0, 33.2, 93.3)	-20° to 80°	-30° to -5°	4.8° to 40°

Table 1 – Parameters used for initialisation search

An image was grabbed every 5-6 seconds, and the images which showed some of the pitch (250 in all) were included in the test. The results are summarised in Table 2 below.

Result	*Number of occurrences*
Successfully initialised (main camera)	195
Successfully initialised (left 18 yard)	7
Successfully initialised (right 18 yard)	6
Total successfully initialised	**208**
Failed (insufficient lines)	31
Failed (not one of these cameras)	8
Failed (lens angle too tight)	3
Failed (other reasons)	0

Table 2 - Results of initialisation test

From these results, it can be seen that the algorithm initialised successfully for 208 out of the 250 images (83% success rate). The majority of the remaining images contained insufficient lines. Many of these were close-ups on particular players – a kind of shot that it is generally unnecessary to augment with virtual graphics. The next most common cause of failure was images coming from a camera other than the three for which the system had been configured. The remaining cause of failure was very tight zoom values (1.8-3.0°), below the minimum value of 4.8° that had been considered worth including in the initial search.

The average total time taken by the initialisation process was 0.99s: 0.02s for locating edge points, 0.33s for computing the spatialised Hough transform (10 spatial subdivisions), and 0.64s for the search (running on a 3.4GHz Pentium 4).

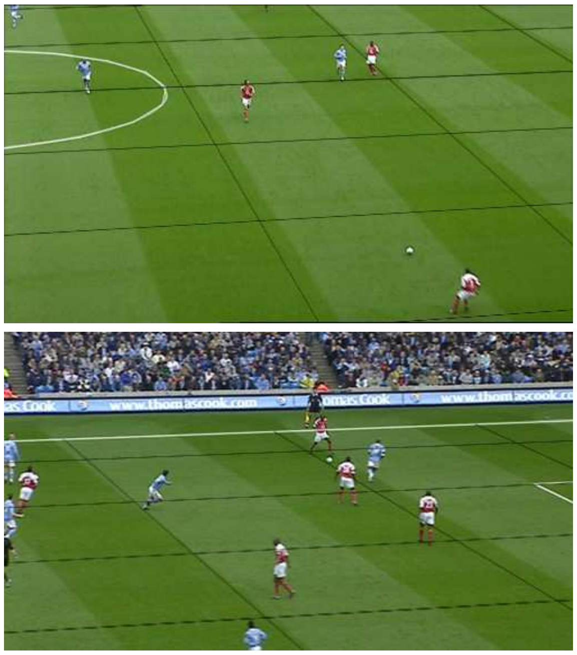


Figure 17 - Some images successfully used for initialisation

Examples of some images containing relatively few lines, that nevertheless the algorithm was able to initialise from, are shown in Figure 17. The lines of the pitch model are overlaid in white; in these images they are very difficult to distinguish from the actual pitch markings, although the line segments used to represent the centre circle are just visible in the upper image. A grid of lines with 10m spacing has been overlaid in black on the ground plane (these should not be confused with the light and dark strips on the pitch itself, which come from the way in which the grass on the pitch was cut, and play no part in the tracking process). The grid overlay is useful when viewing a moving sequence, as it gives an indication of the tracking stability across the whole pitch area.

To illustrate the benefit of retaining spatial information in the Hough transform used during initialisation, the degree to which the correct pose stood out from others was assessed, for several different levels of spatial resolution in the transform.

Figure 18 shows the sum of the bins in the Hough transform corresponding to the expected line positions, for a wide range of pan values, with the position, tilt, roll and field-of-view fixed at values that are approximately correct for the input image shown in the top part of Figure 5. The plot shows how the sum varies as the number of spatial subdivisions in the Hough transform is increased. The peak at around –31° corresponds to the true pan angle. With only one subdivision (i.e. a conventional Hough transform), there are several other significant peaks, and the true peak is only 16% higher than the highest of these. When two subdivisions are used, the true peak is 35% higher than the next-highest peak, and this rises to 100% when 10 subdivisions are used. This indicates the usefulness of subdividing each Hough accumulator in order to retain information on the spatial location of contributing samples.

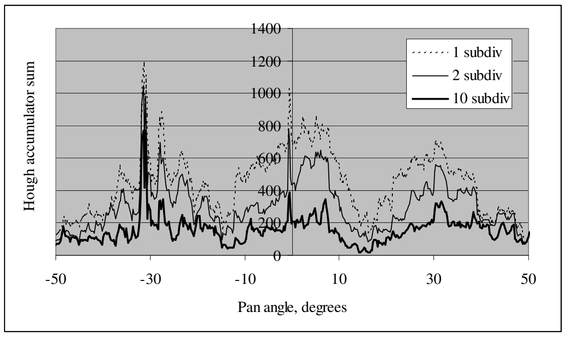


Figure 18 - Sum of Hough accumulator bins as a function of pan angle

5.3 Tracking

An example of the pan and tilt values computed for a 20-second part of the test material is shown in Figure 19. This portion of the sequence showed the main camera panning from the right-hand goal towards the centre of the pitch and back again. Without ground truth data it is difficult to demonstrate the accuracy of the results, but it can be seen from the plot that the values vary smoothly, showing that there is very little random noise present. Note that no filtering or smoothing has been applied.

To obtain a rough measure of the noise present in the camera parameters, the acceleration of the pan angle has been plotted in Figure 20. This would be expected to vary smoothly and to remain close to zero, given the smooth way in which cameras are usually panned. It generally lies within 0.02° of zero, suggesting a noise level of about this amount. The vertical field-of-view was around 11° in this sequence, so this level of pan noise would give rise to movement of the overlaid graphics through a distance of about 1 picture line in a 576-line image. The larger peaks in the acceleration are caused by the centre line coming into or out of view, causing a slight shift in computed pose, due to a small difference between the assumed and actual pitch dimensions.

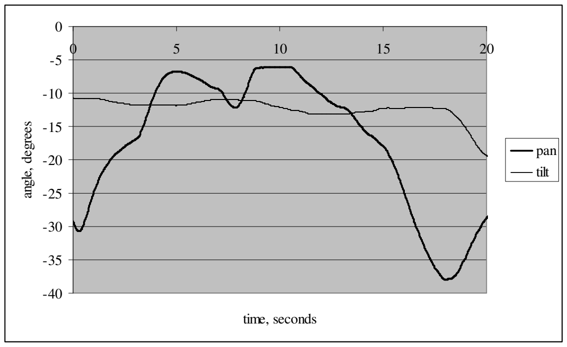


Figure 19 - Pan and tilt angles for a 20s sequence

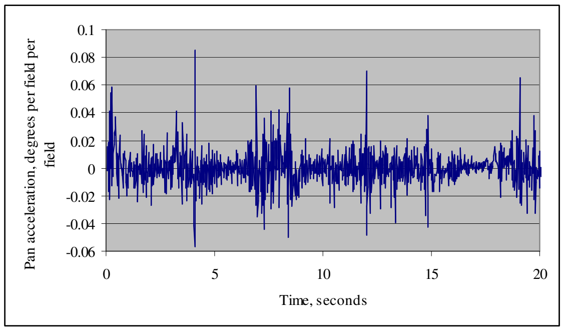


Figure 20 - Second derivative of pan angle

The total processing time for the tracking stage on a typical image was around 5.8ms using a 3.4GHz Pentium 4 processor, although this varied by around ±2ms depending on the number of lines visible. If all detected line points were used in the least-squares optimisation, rather than using the end-points of lines fitted through the points (as discussed in Section 4.2, step 3), the typical processing time rose to around 10.9ms.

5.4 Use with a sports graphics system

The method described here has been incorporated in two commercially-available sports graphics systems [11] [12], in order to allow the system to be used without sensors on the cameras. An example of a graphic overlay generated by the system of [11], using the camera tracking data derived using the method described here, was shown in Figure 1.

5.5 Use in a multi-camera 3D reconstruction system

Another way in which graphics technology can be used to help explain and analyze events during a sports match is by simulating the view from a novel viewpoint. Often, the ideal viewpoint for analyzing a particular incident may be some distance away from where a real camera was placed. Although the system described in [11] incorporates the ability to synthesize a virtual view, it can only use the image from one camera, meaning that the range of virtual viewpoints and directions are limited to those relatively close to this camera. By moving to a multi-camera system, and using all the cameras when reconstructing a 3D model of the scene, a much wider range of viewpoints are possible. Examples of such systems include [5] and [8].

Multi-camera view synthesis requires very accurate calibration of the cameras, in order to ensure correct registration between the views. Reprojection errors should be under 1 pixel. We have found that it is worth using several extensions to the processing method described above in order to improve the accuracy.

First, rather than fitting straight lines through the detected points for each line, then minimizing using the endpoints of the fitted line, we use every detected point. This approach has already been explained in Section 4.2. This allows lens distortion to be estimated.

Secondly, it is worth taking into account the curvature of the pitch: in most cases the pitch slopes down towards the edges in order to improve drainage. Rough observations on a real pitch (at the Millennium Stadium, Cardiff, Wales) suggested that the centre of the pitch was around 0.5m higher in the middle than at the edges. Ideally, the pitch should be surveyed accurately to determine its true shape. Another possibility would be to further extend the method presented in Section 4.2 to solve for the height variation as another unknown in the calibration method, using a model with a small number of parameters to describe the pitch shape. Rather than extending the calibration process itself to do this, we carried out a manual optimization by computing the reprojection error from equation (18) for a range of values for the height at the centre of the pitch. We used a simple one-parameter model to describe the pitch height:

$$y = y_c(1-(\frac{x}{l_x})^2)(1-(\frac{z}{l_z})^2) \qquad (24)$$

where y is the height of the pitch at a point (x, z) on the ground, y_c is the height of the pitch at the centre, and l_x, l_z are the half-length and half-width of the pitch (i.e. the distances from the centre to the edge of the pitch in the x and z directions).

An image of the pitch at the Cardiff Millennium Stadium that was used for an initial investigation is shown in Figure 21. The figure also shows the reprojected pitch model, after calibrating the camera position, orientation, focal length and one distortion coefficient, assuming the pitch is flat. The misalignment between the real and projected pitch lines near the centre of the pitch is about 3 pixels, as shown in Figure 22 (a). This may seem like a small amount, but it is similar in size to the head of the player, and could thus cause the player's head to be significantly degraded (or lost altogether) in a multi-camera 3D reconstruction process. The camera pose computation was repeated using pitch models with assumed heights at the centre of the pitch varying from 0.0 to 0.8m. The RMS reprojection error from equation (18) at each height is plotted in Figure 23. It can be seen that the minimum error occurs for a value around 0.4m. The right half of Figure 22 shows the reprojected pitch line assuming a centre height of 0.4m; the reprojected line is difficult to see as it lies almost on top of the real line.

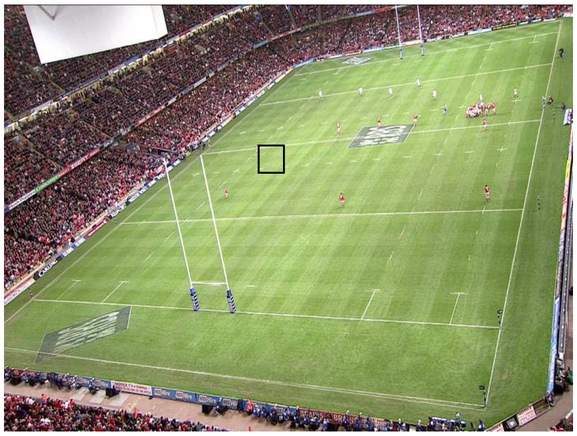


Figure 21 – Flat pitch model fitted to an image from the Cardiff pitch, with a box indicating the region enlarged in Figure 22

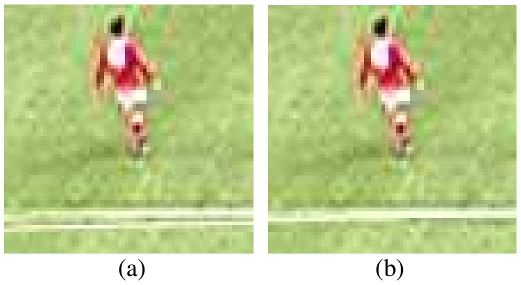


Figure 22 - Close-up showing the improvement in registration with virtual pitch between using a planar pitch model (a) and a curved model (b).

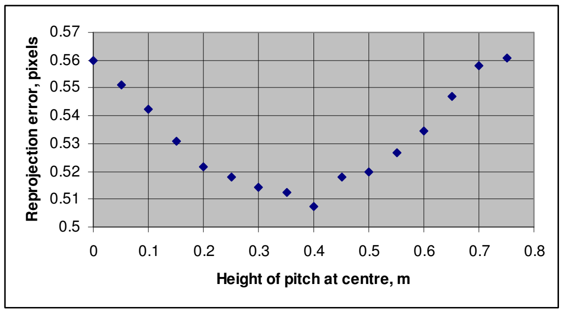


Figure 23 - Reprojection error as a function of pitch model curvature

We have also found it useful to include the goal posts in the pitch model, particularly for close-up shots around the goal where relatively few lines may be visible. Even in wide shots such as that in Figure 21, they can provide an important source of information, as they help to reduce the ambiguities inherent in using a planar calibration scene. When using the line detector to locate goal posts, the test for lines being surrounded by green was disabled, to allow posts to be detected against the crowd.

6. Discussion

The method described here has been found to work well for graphics overlay applications, witnessed by its successful use in two commercial products. It has also been used to calibrate cameras for use in reconstructing 3D models of a football game [5], as described in Section 5.5.

One factor that limits the calibration accuracy is the accuracy to which the overall pitch dimensions are known; as mentioned in Section 1, these vary between grounds. The method presented here could be extended to solve for the pitch width and length, by combining information from multiple images in a manner similar to that described in Section 2, although an independent measurement made with a device such as a laser rangefinder would probably give the most accurate answer.

To improve the robustness and accuracy of tracking when the camera view moves into areas with very few lines, it would be useful to track other image features such as grass texture. This would provide additional information to help changes to pan, tilt and zoom to be estimated, whilst still relying on the appearance of lines at known positions to eliminate long-term drift and provide an absolute scale. An approach based on SLAM (Simultaneous Localisation and Mapping) such as that described in [2] could be used. However, as there is no need to compute the true 3D locations of the additional image features used if the camera position remains fixed, simpler methods of using 2D-3D correspondences are probably more suitable.

To improve the calibration further for applications involving multiple cameras, one approach would be to refine the calibration by using the silhouettes of the players, using an approach similar to that described in [10].

7 Conclusion

This paper has presented an algorithm for the real-time computation of the pose of a camera viewing a scene such as a football match. It includes a method for accurate camera position determination using multiple images, an automatic initialization process to identify the position, orientation and field-of-view of an image from a set of cameras in pre-calibrated positions, and a frame-to-frame tracking method that takes around 6ms per image on a standard PC. It has been successfully used in two commercial sports graphics systems [11] [12], as well as for research into multi-camera 3D reconstruction [5]. Some suggestions for further work to improve the accuracy and robustness have been given.

Acknowledgements

This work was carried out as a part of the EU-funded IST project MATRIS [9]. The author would like to thank his colleagues at BBC Research, BBC Sport, and the team at Red Bee Media Ltd, for helpful discussions and the provision of test data.

References

1. Aldershoff, F., Gevers, T.: Visual Tracking and Localisation of Billboards in Streamed Soccer Matches. Storage and Retrieval Methods and Applications for Multimedia, Proc. SPIE. Vol. 5307, No. 1, pp.408-16, 2004.
2. Davison, A. J.: Real-time simultaneous localisation and mapping with a single camera. Proceedings of the 9th International Conference on Computer Vision, Nice, 2003.
3. Devereux V.G.: Television Animation Store: digital chroma-key and mixer units. BBC Research Department Report No. RD 1984/16.
4. The Football Association.: http://www.thefa.com/TheFA/RulesAndRegulations/FIFALawsOfTheGame/Postings/2002/05/12112.htm
5. Grau O., Hilton, A. et al.: A Free-Viewpoint Video System for Visualisation of Sport Scenes. Proc. of IBC 2006, 7-11 September 2006, Amsterdam, NL.

6. Heather, J. A. Yang, X. D.: Spatial Decomposition of the Hough Transform. Computer and Robot Vision 2005, pp. 476-482.
7. Hough, P.V.C.: Method and Means of Recognizing Complex Patterns, US Patent 3,069,654, 1962.
8. Liberovision AG. http://www.liberovision.com
9. The MATRIS project: www.ist-matris.org
10. Ramanathan, P., Steinbach, E., and Girod, B.: Silhouette-based Multiple-View Camera Calibration. In Proceedings of Vision, Modeling and Visualization 2000.
11. Red Bee Media Ltd.: The Piero Sports Graphics system www.redbeemedia.co.uk/piero, www.bbc.co.uk/rd/projects/virtual/piero/
12. RT Software Ltd.: The Tog Sports system. www.rtsw.co.uk.
13. Thomas, G. A.: Real-time Camera Pose Estimation for Augmenting Sports Scenes. Conference on Visual Media Production (CVMP), London, 29-30th November 2006.
14. Tsai, R. Y.: A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Journal of Robotics Automation, pages 323-344, Vol. RA-3, No. 4 1987.
15. Vacchetti, L., Lepetit, V., Fua, P.: Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. International Symposium on Mixed and Augmented Reality, pp. 48-56, 2004.
16. Watanabe, T., Haseyama, M., Kitajima, H.: A soccer field tracking method with wire frame model from TV images. 2004 International Conference on Image Processing, pp. 1633-1636.
17. Yu, X., Leong, H.W., Xu, C., Tian, Q.: A Robust Hough-Based Algorithm for Partial Ellipse Detection in Broadcast Soccer Video. 2004 IEEE International Conference on Multimedia and Expo, pp. 1555-1558.
18. Zhang, Z.: Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. Seventh International Conference on Computer Vision (ICCV'99) - Volume 1, p. 666, 1999.