Multi-view 3D Position Estimation of Sports Players

Robbie Vos* and Willie Brink
Applied Mathematics
Department of Mathematical Sciences
University of Stellenbosch, South Africa
*Email: vosrobbie@gmail.com

***Abstract*—The problem of estimating the positions of players on a sports field using multiple cameras is considered. A hierarchical particle filter is used to track the players through each video sequence. Position estimation is then performed using multi-view triangulation. We also introduce a feedback loop whereby 3D data can be fed back to the 2D trackers to correct errors. Experiments suggest that the system performs well and yields high accuracy for position estimation, with an average tracking error less than 10cm.**

I. Introduction

Millions of people around the world follow sports in some form or another. With all the interest that sport gathers, spectators and fans are increasingly looking for up-to-date statistics on all aspects of their game of choice. Much of these statistics are manually extracted while watching the game or from video footage after the game has completed.

A system that is able to track players on a field during a game will be able to automatically provide a range of statistics that is relevant for analyzing player and team performances. It will be possible to calculate how much distance players are covering in a match as well as which areas of the field they spend the majority of time. This can be used to measure the work rate of different players on the field or to compare tactical strategies between teams.

Some work in this field has been done by previous researches. Khan et al. [1] tracked people moving in a building by comparing the movement of people with camera field of view lines. Cai et al. [2] also looked at tracking people in multiple views. People are detected by segmenting the image into foreground and background, building a model of the background and comparing the current view to the model. Foreground regions are then analyzed to detect human shapes and people are tracked by feature points. While both of these systems are able to track people in 2D and perform matches between the cameras they lack some functionality that is required for 3D tracking. The biggest problem is that neither of the two solutions are able to calculate the 3D position of a tracked person. Another problem with the two systems is that they are unable to track people through occlusions.

A full multi-view 3D tracking systems was developed by Alahi et al. [3] for tracking players on a basketball court. Using adaptive mixture models a foreground image is extracted for each camera. Foreground silhouettes for each camera are projected onto the ground plane and players are tracked by matching ground plane projections and modelling player behaviour.

Another multi-view 3D tracking technique was developed by Xu et al. [4]. A mask of the field is extracted using background modelling techniques, and used to detect players while excluding unwanted regions. Kalman filtering is used for both 2D and 3D tracking.

In our solution motion detection was found to be a fast and effective means of detecting players. In order to track players in each camera view a fast hierarchical approach to the particle filter is chosen due to it robustness to common tracking problems. Finally player positions are found using multi-view triangulation.

The rest of this paper is structured as follows. Section II explains the camera calibration required for the system to function. In section III 2D player tracking is discussed. Estimation of player positions is considered in section IV and the paper concludes with some results in section V and conclusions in section VI.

II. Calibration Method

In order to perform position estimation of players on a field the cameras used need to be calibrated to the world around them. This calibration can be split into two sections: internal and external calibration.

During internal calibration the parameters pertaining to the camera itself are calculated. These parameters remain constant for the camera (the focal length may change, but is assumed to remain constant) and as such may be calculated before the system is deployed. A popular approach to perform this calibration is to use a checker-board pattern where the number and physical size of squares are known. The corners of each of the squares on the checker-board can either be detected automatically or selected by a human. Calibration can then be performed by using the corners as interest points, with a method such as the one developed by Tsai [5].

For external calibration several point correspondences are needed between points with known real-world coordinates and the image coordinates of those points. A minimum of three such point correspondences is required and any points may be used with the restriction that at least three points do not lie on a single line in space.

A convenient set of points to use is the set of corners made by intersecting lines on the field. These lines can be accurately detected and the corresponding intersections found to give the

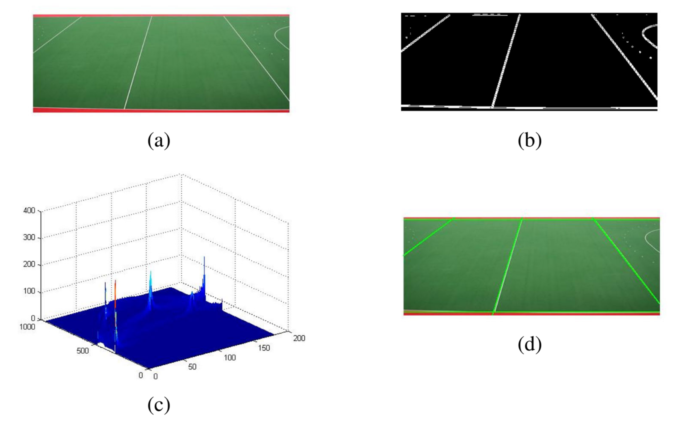

Fig. 1. Illustration of the Hough transform for line detection: (a) original image, (b) Sobel edge detection, (c) Hough transform histogram and (d) detected lines.

corners. To detect the lines, and through them the field corners, the Hough transform is used. Figure 1 illustrates. Once the lines are found the intersection of those lines can be found by turning to homogeneous coordinates. The standard form for lines in $\mathbb{R}^2$ is $ax + by + c = 0$, allowing for each line to be uniquely represented by a coefficient vector $\mathbf{l} = [a, b, c]^T$. The intersection, $\mathbf{p}$, of any two lines represented this way is simply the cross-product between the two lines:

$$\mathbf{p} = \mathbf{l} \times \mathbf{l}'. \tag{1}$$

As the real-world coordinates for the corners are known (if the dimensions of the field are known) they make ideal candidates for calibration points. The calibration of the external parameters can then be done using the work presented in [6].

III. Tracking in 2D

Tracking players in 3D through the use of multiple cameras first requires that the players be tracked in each of the individual camera sequences. To accomplish this a hierarchical approach to the particle filter is used. In this approach objects are represented using several different descriptors. Descriptors vary from coarse, fast descriptors that may yield many false positives, to fine but slow descriptors for more precise classification. Objects are first compared to the coarse descriptors, and if they are a good match are then passed to the slower second stage. This hierarchical approach allows for much faster execution of the filter while maintaining good results.

A. First Stage Descriptors

For first stage descriptors in the hierarchical particle filter the rectangle features of Viola and Jones [7] are used, specifically those depicted in Figure 2. The descriptors act similar to a mask that is convolved with the image. At each pixel location the intensity values of pixels in the region around the object are added and subtracted. Figure 2 and the following equation illustrate how the features are calculated using a 9×9 block (in our system the block size is closer to 80×40),

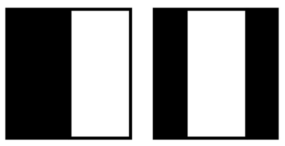

Fig. 2. Illustration of rectangle features used as a tracking descriptor.

$$R(i,j) = \sum_{j=-4}^{4} \left[\sum_{i=-4}^{-3} I(i,j) + \sum_{i=3}^{4} I(i,j) - \sum_{i=-2}^{2} I(i,j) \right]. \tag{2}$$

These features are chosen as they are very fast to calculate. Using the integral image [7] further speed increases can be made. The integral image is such that each pixel value is the sum of all the intensity values of all pixels to the left or above the current pixel, and allows extremely fast calculation of the sum over an arbitrary blocks of pixels in the image.

Two rectangle features are calculated for every particle. The dissimilarity measure between the calculated feature and the model feature is taken as the absolute difference between the two. Particle weights for each feature can now be calculated. Each particle has two first-stage weights assigned to it, one for each feature. The weights can be combined either using the average of the two or by multiplying the two together. An average of the two will give particles that have a good match in both features a high final weight, however particles that have one good and one poor match will still have relatively high scores. Only particles that score a good match in both features are of interest, making the multiplication method better suited to the problem.

B. Second Stage Descriptors

In the second stage of the hierarchical particle filter only those particles that have a contributable weight after the first stage, i.e. those that have a first-stage weight greater than some threshold, are considered. Particles with small weights after the first stage are considered to be very different to the object, and by ignoring them in the slower second stage we gain computational time. In this stage a histogram of oriented gradients is calculated as a more precise descriptor than the rectangle features used in the first stage.

The calculation of a histogram of oriented gradients requires gradient vectors for each pixel in the image. Discrete derivative operators, such as the Sobel operators, can be used for this purpose and yield two edge images: E_h that highlights horizontal edges and E_v that highlights vertical edges. The magnitude and angle of the gradient vector at each pixel is then calculated as

$$M(i,j) = \sqrt{E_h(i,j)^2 + E_v(i,j)^2}, \tag{3}$$
$$G(i,j) = \arctan\left[E_v(i,j)/E_h(i,j)\right]. \tag{4}$$

Gradients with a magnitude greater than some threshold are then binned into a histogram according to their angles.

The histograms of all the particles need to be compared with that of the model in order to arrive at some dissimilarity value. There are various ways in which the distance between two histograms can be calculated.

The "city block" and Euclidean distances (i.e. the L_1 and L_2 norms) are fast to compute but do not perform adequately on histograms where the order of the bins carry some meaning. Consider, for example, three histograms $h_1 = [1\ 5\ 1\ 1\ 1\ 1\ 1]$, $h_2 = [3\ 1\ 3\ 1\ 1\ 1\ 1]$ and $h_3 = [1\ 1\ 1\ 1\ 3\ 3\ 1]$. Here h_1 and h_2 should be considered as being much closer to one another than, say, h_1 and h_3. However the Euclidean distance gives $d(h_1, h_2) = d(h_1, h_3) = \sqrt{24}$.

Distances that measure the difference between discrete probability density functions can also be used to compare histograms. Examples include the Kullback-Leibler divergence and the Bhattacharyya distance. These measures, however, also fail for the same reason as the L_1 and L_2 norms.

There are more indicative measures of the distance between histograms. The earth mover's distance (EMD) [8], for example, regards the histograms as piles of dirt and determines the minimum cost required to turn one into the other (where cost is defined as amount of dirt times the distance by which it is moved). This optimization problem, although linear, is rather computationally intensive for the purposes of this problem. Cha and Srihari [9] proposed a measure which is related to the EMD but is much faster to calculate. Because gradient orientations range between 0° and 360°, with the endpoints regarded as equal, the modulo distance measure (as explained in full detail in [9]) is used.

Particles that were ignored in the second stage due to low first stage weightings still require a second stage weight for the filter to propagate forward. As there is no distance measure calculated for these particles, they are given a second stage dissimilarity value equal to twice the largest value calculated in the second stage. The second stage weights for all the particles can now be calculated.

C. Filter Output and Updating the Filter Model

Once all the first and second stage weights have been calculated a filter output can be obtained. The first and second stage weights for each particle are multiplied together, after which all the weights are normalized to produce a final weight for each particle. A weighted average of all the particles is taken to find the filter output X:

$$X(x, y) = \sum_{i=0}^{n} w_i p_i(x, y). \tag{5}$$

The model that is being tracked must now be updated for the next iteration of the filter. After finding X the first as second stage descriptors are calculated around that point, and those descriptors are used for the next iteration of the filter.

The next section looks at combining the data from the 2D trackers to estimate 3D position and track players in real-world coordinates.

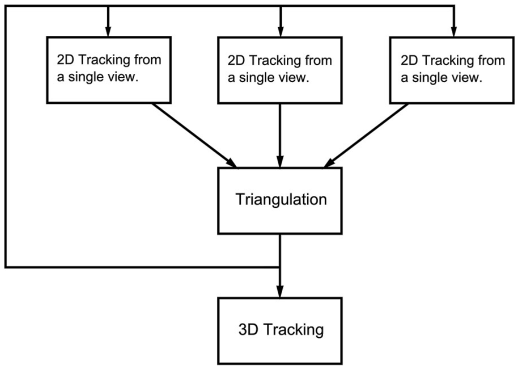
2D Tracking from a single view.
2D Tracking from a single view.
2D Tracking from a single view.
Triangulation
3D Tracking

Fig. 3. Tracking feedback loop: 2D data is used to calculate 3D points which are fed back to the 2D trackers for error correction.

IV. 3D Triangulation

Once players are successfully tracked in 2D it becomes possible to estimate and track their 3D positions. To accomplish 3D tracking the data from the 2D tracking is required for each player in each view. This 3D data can then be fed back to the 2D trackers to check and possibly correct errors in the 2D tracking. This forms a feedback loop as illustrated in Figure 3.

The rest of this section details the processes of combining the various views, triangulating the player positions and the feedback loop.

A. Matching Players Between Views

Before triangulation of player positions is possible it is necessary to match the players between views. Several options exist when trying to accomplish this, such as shape, colour and position.

Shape and colour methods operate by quantifying the shape and/or colour aspects of the person being tracked using, for example, edge or colour histograms. These histograms can be compared to histograms of players being tracked in different views, and should a match be found they are assumed to be the same player in the different views. These methods can fail, however, when applied to the problem of tracking sports players. Colour methods are ineffective as players on the same team will all be wearing similar clothing. Attempting to match players in this situation will result in multiple matches making it impossible to know which is the correct match. Shape matching on the other hand fails as the player shape may vary drastically when viewed from different angles.

Position matching estimates the position of the player on the field from each view individually. This estimation may not be highly accurate, but it does allow one to identify clusters of estimated points. These points may indicate the presence of a player on the field and the corresponding projections of the players in the 2D views.

The single view estimation begins by finding the line in 3D passing through the tracked point on the image plane. Once this line has been found the intersection between the line and a plane some distance above the field is calculated (according to [10] the average height of a male in South Africa is 168 cm, indicating a plane 84 cm above the playing field should be chosen when tracking the center of the player, while the average height of a female in South Africa is 158 cm indicating a plane 79 cm above the playing field).

After the intersection points have been calculated they can be compared to intersection points from different views. If clusters of intersection points are found close to one another then a match is made between the different views. If two or more intersection points from a single view are located close to each other, i.e. during an occlusion, that location cannot be assigned with a high level of certainty and it is ignored until the two tracked players move away from one another. Also note that once a match has been made, that match remains for the rest of the program execution and the matching step does not need to be repeated.

Another advantage of this matching method is that the calculation of the 3D lines is also required for the triangulation step. This has the effect that the matching step does not greatly increase computational time.

B. Triangulating Player Positions

Once all the players are tracked in each of the video sequences the positions of players can be triangulated on the field. Two options exist for triangulating from multiple views:

- pairwise, back-projection error minimization;
- multi-view, forward-projection error minimization.

In the first case the object is triangulated for each possible pair of cameras using back-projection error minimization technique. This will give $\frac{1}{2}n(n-1)$ solutions when the object is visible by n cameras. These solutions may then be combined to get a final point by taking the average or least-squares of the set of points. The second option triangulates a single point using all views in a single step, by minimizing the forward projection error rather than the backward projection error.

Testing of the two techniques produced similar results, causing a decision between the two to hinge on computational speed. In this respect the multi-view triangulation is superior to pairwise triangulation due to the fact that multi-view triangulation increases linearly in computational complexity with the number of cameras while pairwise triangulation is of order n^2.

Triangulation from multiple views presents new challenges, but also some benefits above two-view triangulation. On the one hand multiple views provide more information, allowing for more accurate triangulation. On the other hand it is harder to combine the data in a computationally inexpensive manner while keeping a high degree of accuracy. We try to minimize the projection error e_p.

The first step in minimizing e_p is to find the projection of the point $\mathbf{X}$ on each of the lines $\mathbf{l}_i$. Each of the lines $\mathbf{l}_i$ can be written as $\mathbf{l}_i = \mathbf{p}_i + k\mathbf{n}_i$ where $\mathbf{p}_i$ is a point on the line and $\mathbf{n}_i$ is a unit vector in the direction of the line. To solve for each of the lines one begins with the camera equation:

$$\mathbf{x} = \mathbf{KR}[\mathbf{I}| - \tilde{\mathbf{C}}]\mathbf{X} = \mathbf{KR}\tilde{\mathbf{X}} - \mathbf{KR}\tilde{\mathbf{C}} \tag{6}$$

which can be rewritten as

$$\tilde{\mathbf{X}} = \mathbf{R}^T\mathbf{K}^{-1}\mathbf{x} + \tilde{\mathbf{C}}. \tag{7}$$

Using the camera center, $(0,0,0)^T$, and the point on the image plane through which the line is to be drawn, $(x,y,1)^T$, and substituting for $\mathbf{x}$ in equation (7) two points, $\mathbf{q}_1$ and $\mathbf{q}_2$, in the real-world coordinate system emerge that both lie on the desired line. It is now possible to solve for $\mathbf{p}_i$ and $\mathbf{n}_i$:

$$\mathbf{p}_i = \mathbf{q}_1, \quad \mathbf{n}_i = \frac{\mathbf{q}_1 - \mathbf{q}_2}{||\mathbf{q}_1 - \mathbf{q}_2||}. \tag{8}$$

The projection, $\mathbf{y}_i$, of $\mathbf{x}$ on $\mathbf{l}_i$ can then be found as

$$\mathbf{y}_i = \mathbf{n}_i\mathbf{n}_i^T(\mathbf{x} - \mathbf{p}_i) + \mathbf{p}_i, \tag{9}$$

with the total projection error

$$e_p = \sum_i \|\mathbf{y}_i - \mathbf{x}\|^2. \tag{10}$$

We now want to find $\mathbf{x}$ that minimizes e_p:

$$\begin{aligned} e_p &= \sum_i \|\mathbf{n}_i\mathbf{n}_i^T(\mathbf{x} - \mathbf{p}_i) + \mathbf{p}_i - \mathbf{x}\|^2 \\ &= \sum_i \|(\mathbf{n}_i\mathbf{n}_i^T - \mathbf{I})\mathbf{x} - (\mathbf{n}_i\mathbf{n}_i^T - \mathbf{I})\mathbf{p}_i\|^2 \\ &= \sum_i \|\mathbf{A}_i\mathbf{x} - \mathbf{b}_i\|^2 \\ &= \sum_i (\mathbf{A}_i\mathbf{x} - \mathbf{b}_i)^T(\mathbf{A}_i\mathbf{x} - \mathbf{b}_i). \end{aligned} \tag{11}$$

Taking the derivative of (11) with respect to $\mathbf{x}$ and setting it equal to zero yields

$$\begin{aligned} \frac{\partial e_p}{\partial \mathbf{x}} = \sum_i \left[2(\mathbf{A}_i^T\mathbf{A}_i)\mathbf{x} - 2\mathbf{A}_i^T\mathbf{b}_i\right] &= \mathbf{0} \\ \sum_i (\mathbf{A}_i^T\mathbf{A}_i)\mathbf{x} &= \sum_i \mathbf{A}_i^T\mathbf{b}_i \\ \left(\sum_i \mathbf{A}_i^T\mathbf{A}_i\right)\mathbf{x} &= \sum_i \mathbf{A}_i^T\mathbf{b}_i. \end{aligned} \tag{12}$$

Equation (12) is now in a familiar form, $\mathbf{Cx} = \mathbf{d}$, allowing us to solve it using standard linear algebra techniques.

C. Error Correction

When tracking the 3D positions of players using multiple cameras, this 3D data can be used to increase the accuracy of the tracking in the individual camera scenes. By comparing the 3D position obtained by triangulation to an estimation of the position based only on each view individually it becomes possible to detect and correct errors in the single view tracking.

After triangulating a player based on the 2D tracking results a set of distance measures can be calculated between the triangulated point and the projected point from each camera, using the Euclidean distance. This projected point is the same point as calculated in section IV-A when finding player matches.

If any of the distance measures are greater than some threshold it may indicate that there is a problem with the

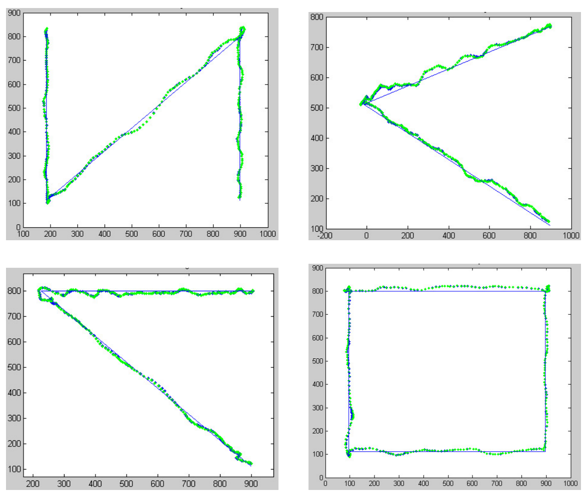

Fig. 4. Triangulation of two image sequences using forward (green dots) and back projection (blue dots) methods. The solid blue line indicates the ground truth.

corresponding 2D tracking. To correct this error the player's location is triangulated a second time, using only tracking results from those trackers where the distance measure is below the threshold. This new 3D point, $\mathbf{X}$, is then projected back to the discredited views using the standard camera equation:

$$\mathbf{x} = \mathbf{P}\mathbf{X}. \tag{13}$$

The tracker corresponding to that player in that view can then be restarted at the calculated point $\mathbf{x}$. If less than two of the projection points fall within the threshold then there is no reliable way to determine which of the trackers have failed and which of them are still accurate. In this case the player may need to be dropped from 3D tracking and all corresponding 2D trackers stopped. The player will then be detected and tracked again as a new entity as if it is a new player on the field.

V. Experiments

To test the system that was developed several test were completed. The first test was to measure the accuracy of the 3D position estimation of a single player running on a field. Figure 4 shows the results for four such sequences. The solid blue line in each figure is the ground truth path that the humanoid ran over, as viewed from above. The blue points are the back projection results and the green points are the forward projection results.

This test indicates that triangulation results for the two methods are similar. This test also indicates that the proposed tracking and triangulation method succeeds at locating a player on the field of play. In the given figures one unit of measurement corresponds to 2 cm on a real-life field. The maximum deviation from the ground truth between the four sequences is 20 units (40 cm) while the average deviation is about 5 units (10 cm).

In Figure 5 the results of the full system can be seen for tracking 4 players as seen in 4 views over 286 frames

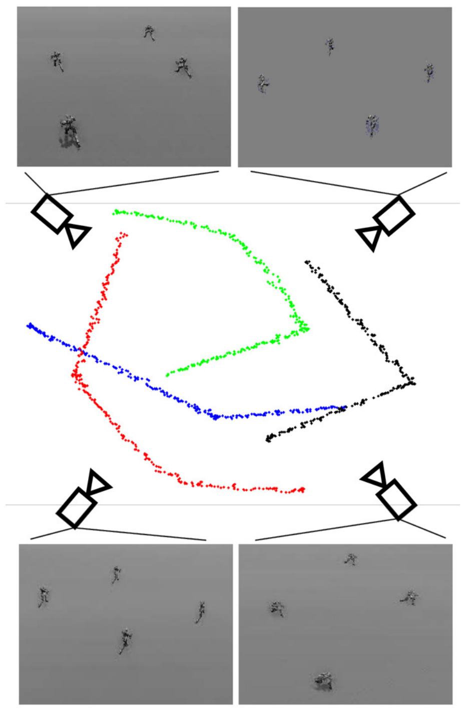

Fig. 5. Results for tracking four players with four views though 286 frames.

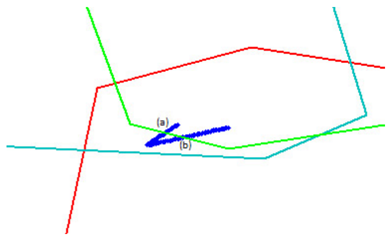

Fig. 6. Tracking a player crossing the field of view line of a camera, shown here from a top down view.

(roughly 10 seconds). The coloured dots indicated the triangulated position for each player through the sequence. The approximate position of the cameras are shown by the little camera drawings. As can be seen from this figure the system as a whole functions as desired: detecting, tracking and triangulating each of the players. This is, however, an ideal case and further testing of some possible problem scenarios needs to be done.

During full system testing two cases of interest were identified. The first case is when a player leaves or enters a camera field of view and the second is when a 2D tracker loses its player due to occlusion.

In Figure 6 the solid lines indicates field of view boundaries of the different cameras, and the blue dots indicates the path followed by the player. At point **(a)** the player moves out of the view of the camera indicated by the green lines. At this point the corresponding 2D tracker is stopped and the player is triangulating with the remaining views. At point **(b)** the

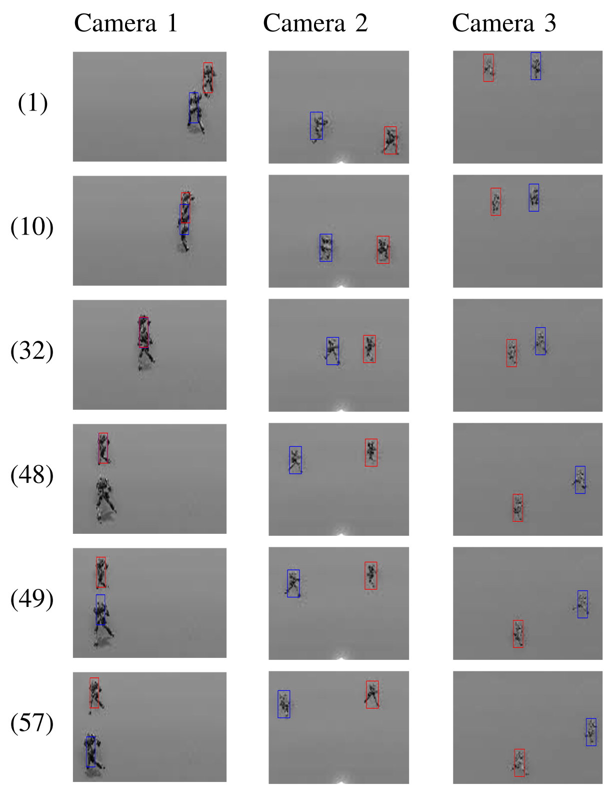

Fig. 7. Automatic correction of tracking occlusion. Frame numbers are listed on the left of the images.

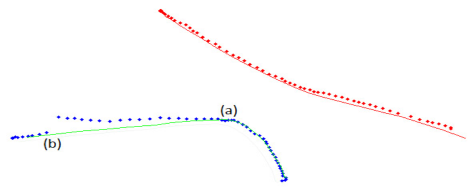

Fig. 8. Triangulation results of the multi-view tracking in Figure 7, as viewed from above.

player again moves back into the field of view of the camera and is then again tracked in that view. Single view position estimation showed that this view corresponded to the player already being tracked the player is then triangulated using the data from that view as well.

The second case is illustrated in Figure 7, where in one view the two players move in such a way that the one player occludes the other whilst in the other two views they move apart from each other. In frame (1) the players are a distance away from each other. By frame (10) they have started to occlude each other and at frame (32) they are heavily occluded. As can be seen at frame (48) the tracker indicated by the blue square has begun to track the incorrect player. At this point the system detected that the 2D tracker has lost the player and moved from the correct path and attempts to correct the mistake. In frame (49) the 2D tracker in the first view has been corrected by back projection after triangulating the player using the rest of the views. By frame (57) the 2D tracker has corrected itself and is tracking the player correctly again.

The plot of the players' positions in Figure 8 illustrates the effect of this occlusion. At point **(a)** the triangulation result begins to drift from the ground truth line. At point **(b)** the 2D tracker is corrected and the triangulation results snap back to the correct ground truth line.

VI. Conclusions and Future Work

In this paper the problem of estimating the 3D position of players on a sports field using multiple cameras was discussed. A system was developed, using motion detection to find players, a fast hierarchical particle filter to track players in 2D, and multi-view triangulation to find player positions.

The results obtained for the system were very promising overall. While some improvements can be made the system is able to solve the initial problem to a satisfactory extent. Occlusions present some problems, however the use of multiple cameras goes some way to solve this.

Some future work that may improve the system would be to remove the fixed block size used in 2D tracking. This may allow cameras to view larger areas of the field as players do not need to appear in some specific size in the image. Using a player recognition algorithm may also improve both 2D and 3D tracking results. Trackers can correctly distinguish between players after occlusions by recognising the player they need to track. More precise matching of players between views may also be possible if recognition is used rather than just position matching.

VII. Acknowledgments

We thank MIH and the National Research Foundation for financial assistance.

References

[1] S. Khan, O. Javed, Z. Rasheed, M. Shah, "Human tracking in multiple cameras", IEEE International Conference on Computer Vision, vol. 1, pp. 331–336, 2001.
[2] Q. Cai, J. Aggarwal, "Tracking human motion using multiple cameras", International Conference on Pattern Recognition, vol. 3, pp. 68–72, 1996.
[3] A. Alahi, Y. Boursier, L. Jacques, P. Vandergheynst, "Sport players detection and tracking with a mixed network of planar and omnidirectional cameras", IEEE International Conference on Distributed Smart Cameras, pp. 1–8, 2009.
[4] M. Xu, J. Orwell, L. Lowey, D. Thirde, "Architecture and algorithms for tracking football players with multiple cameras", Intelligent Distributed Surveilliance Systems, pp. 51–55, 2004.
[5] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses", IEEE Journal of Robotics and Automation, vol. 4, pp. 323–344, 1987.
[6] M. Haralick, C. Lee, K. Ottenberg, M. Nolle, "Review and analysis of solutions of the three point perspective pose estimation problem", International Journal of Computer Vision, pp. 592–598, 1991.
[7] P. Viola, M. J. Jones, "Rapid object detection using a boosted cascade of simple features", IEEE Computer Vision and Pattern Recognition, vol. 1, pp. 511–518, 2001.
[8] Y. Rubner, C. Tomasi, L. J. Guibas, "A metric for distributions with applications to image databases", International Conference on Computer Vision, pp. 59–66, 1998.
[9] S. Cha, S. Srihari, "On measuring the distance between histograms", Pattern Recognition, vol. 35, pp. 1355–1370, 2002.
[10] http://www.doh.gov.za/facts/1998/sadhs98/chapter13.pdf