Combining motion detection and hierarchical particle filter tracking in a multi-player sports environment

Robbie Vos, Willie Brink

Department of Mathematical Sciences
University of Stellenbosch, South Africa
vosrobbie@gmail.com, wbrink@sun.ac.za

Abstract

We consider the problem of detecting and tracking players on a sports field in a grayscale video sequence from a fixed camera. Motion detection is performed as a separate subsystem, to locate moving objects and pass them to a tracker, and involves background modelling by fast rectangle features and blob detection. A particle filter is utilized for the tracking of objects, by a fast hierarchical approach, based on rectangle features and histograms of oriented gradients. Experiments suggest that the system as a whole performs well and may also be fast enough for real-time applications. Problems do occur from time to time but mainly due to challenging occlusions.

1. Introduction

Accurate detection and tracking of humans in a video sequence from a single fixed camera are of great interest to a wide range of fields — security and sports applications being two principal examples. In this paper we consider the problem of detecting and tracking a number of moving people simultaneously, specifically in a multi-player sports environment like soccer or field hockey. Successful detection and tracking would allow for the extraction of statistical data related to, for example, distances covered by individual players and where on the field they spend time. It might also serve as a useful input for a more sophisticated statistical analysis of tactics and strategies adopted by the different teams.

The problem of tracking sports players, as opposed to say pedestrians on the street, poses many challenges. The movement and variation in pose of a sports player is often complex, quick and abrupt, and therefore hard to predict. Interaction among the players in a sports environment often lead to perplexing occlusions from the camera's point of view, and may cause a substantial amount of uncertainty in the identification of individuals.

On the other hand, a sports environment also places some useful restrictions on the general problem of human detection and tracking. The background is usually fairly uncluttered and predictable and, because players are restricted to the field, the camera can be calibrated to the scene beforehand and fixed for the duration of a game.

If statistical analysis of a game in its entirety is the only objective it may suffice to only capture the necessary video data during the game and then perform tracking off-line at a later stage. A system functioning in real-time, however, facilitates things like online analysis, critical event detection that may require immediate action, live broadcast annotations and even automatic commentary. For these reasons we regard real-time execution as a high priority.

In this work we are also interested in tracking players specifically in grayscale (as opposed to colour) video sequences. This type of input not only implies cheaper hardware but also poses an interesting question from a research point of view: can sports players be detected and tracked successfully without the use of colour cues?

The task of locating objects, in this case sports players, in a video can be divided into two parts: detection and tracking. Here we provide a short summary of some of the main techniques for each.

Much work has been conducted in the area of automatic human detection from images and video. Techniques include supervised learning approaches that match object and non-object templates [1], feature-based methods such as scale-invariant features [2] and histograms of oriented gradients [3], and matching of parts methods [4]. These and other techniques are known to produce good results but are, in general, rather slow. Applicability is also often limited to the detection of pedestrians on the street and therefore allow only small variations in appearance and pose.

We opted for a motion detection approach as it tends to be much faster than those methods that detect by recognition. Essentially, moving objects in a current video frame are isolated by comparison with some model of the background [5, 6, 7]. A trained classifier can then decide which of these moving objects are in fact humans, a task that should be relatively straightforward given the constraints in a sports environment. The accuracy of motion detection may not always be on par with that of object detection methods, but in a video sequence it is usually not of utmost importance that a player be detected at the first possible moment. This last point is particularly valid when computation speed is important.

The second part, after detection, is tracking. A good overview of different tracking approaches developed recently is given in [8]. A popular choice is the use of particle filters, for their versatility, speed and robustness. Song *et al.* [9] combine the particle filter with probabilistic detection, while Yang *et al.* [10] take a hierarchical approach with edge and colour features.

For the tracking part of this work we adopt ideas from [10] but, since colour is not available, we rather use some rectangle features (akin to those used by Viola and Jones [11]) and edge orientation histograms.

The rest of the paper is structured as follows. Section 2 discusses our approach to motion detection and the link between the motion detector and the tracking system. Section 3 provides a brief overview of the particle filter and a discussion of how it is used. Some experimental results are given in section 4 and we conclude in section 5.

2. Motion detection

A popular approach for performing motion detection in a video sequence is to compare the current frame to some model of the static background, in an attempt to isolate moving objects. This procedure is often referred to as "background subtraction".

The procedure outlined in this section performs motion detection in three stages: (i) a difference image is first created by means of background subtraction; (ii) regions of abundant motion are then extracted; and (iii) those regions are analyzed and, if deemed valid, passed to the tracking subsystem.

2.1. Difference image

The first step in the motion detection process is background subtraction. A model of the background is needed for this purpose, to which the current frame can be compared. Pixel regions that differ substantially from the background model are then interpreted as motion. Although sophisticated background modelling techniques are available, a notable example being one that does it by a mixture of Gaussians [6], a main objective of ours is speed and we therefore opt for a simpler approach that turns out to be highly effective.

For the construction of a background model we utilize rectangle features [11] and average them over a number of previous frames. We use specifically the 2-rectangle feature depicted in Figure 1(a). Let $I_s(i,j)$ denote the grayscale intensity of the pixel at row i and column j in frame s of the sequence. Rectangle features can be computed for that frame to give a feature image F_s, where

$$\begin{aligned} F_s(i,j) &= I_s(i,j) + I_s(i,j+1) \\ &\quad - I_s(i+1,j) - I_s(i+1,j+1). \end{aligned} \tag{1}$$

We experimented with different choices of rectangle features (horizontal, vertical, diagonal, etc.), and found the vertical one shown in Figure 1(a) to be most successful. It can of course be explained by the fact that, if the camera is not tilted, people tend to exhibit more vertical edges (if standing up fairly straight) and move more in a horizontal direction across the frame.

The background model M_k associated with the current frame k is determined as a weighted average of the feature images of the previous p frames, i.e.

$$M_k(i,j) = \sum_{s=k-p}^{k-1} w_s \, F_s(i,j), \tag{2}$$

where p is the memory length of the model. The value of p should be chosen with the background situation in mind: if the background remains static p can be increased to give a more accurate model, while a smaller value of p would imply faster updating of the model. The symbol w_s in equation (2) indicates some weight attributed to frame s. We assume here without loss of generality that $w_{k-p} + w_{k-p+1} + \ldots + w_{k-1} = 1$. The weights may be chosen constant, in which case all frames carry the same importance, or chosen such that $w_{k-p} < w_{k-p+1} < \ldots < w_{k-1}$ to facilitate a "fading memory".

(a) (b) (c)

Figure 1: *A depiction of (a) the 2-rectangle feature, and (b) & (c) some 3-rectangle features. An image is convolved with one of these masks. Black indicates regions of negative elements and gray regions of positive elements in the mask.*

Note that the background model is constantly updated, in fact at every frame, to accommodate variations in light intensity, unforeseen camera motion, and motion due to non-player objects such as trees in the background or spectators. The rectangle features are extremely quick to determine, so that this constant updating of the background model is by no means a computational bottleneck.

After specifying a background model M_k for the current frame k, a difference image D_k is created simply as the absolute difference between F_k and M_k, i.e.

$$D_k(i,j) = |F_k(i,j) - M_k(i,j)|. \tag{3}$$

Every value in D_k is interpreted as a sort of likelihood of motion occurring at the corresponding pixel. Figure 2 shows an example frame and its difference image (normalized for display purposes). Observe that the shadows of the players are hardly visible in the difference image. This is another exceptionally useful effect of the chosen rectangle features. The background model consists of relative pixel differences so that, for example, grass in sunlight and grass under a cast shadow are viewed as being equal (or at least close to being equal).

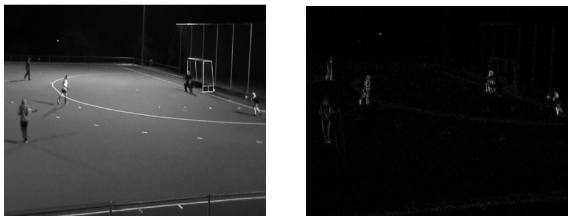

Figure 2: *One frame from a grayscale video sequence and the difference image resulting from our background subtraction procedure. Note that shadows hardly appear in the difference image.*

2.2. Extracting regions of motion

In the next stage we search for sizable regions of high density in the difference image that would appear as a result of objects (hopefully players) that move against the background.

Firstly the difference image is convolved with an averaging filter of some specified size $(2a+1) \times (2b+1)$. It yields what we refer to as the motion image N_k associated with frame k, such that

$$N_k(i,j) = \frac{1}{(2a+1)(2b+1)} \sum_{s=i-a}^{i+a} \sum_{t=j-b}^{j+b} D_k(s,t). \tag{4}$$

This averaging process serves two purposes. Firstly the motion values of individual pixels are summed over a neighbourhood so that regions with high motion density stand out. Secondly regions containing a small amount of motion, likely due to noise, are flattened. Figure 3 shows on the left the motion image corresponding to the frame in Figure 2. Regions containing a large amount of motion are clearly visible.

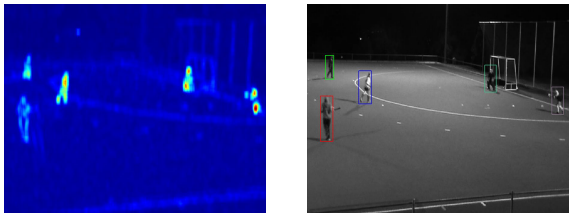

Figure 3: *A visualization of the motion image corresponding to the frame in the previous figure (left), and objects extracted by the detection of dense motion blobs (right).*

Objects can now be extracted from the motion image by, for example, a blob detection method. A fast and easy technique for isolating different objects, that we found to be very successful, is to simply threshold the motion image by some value t, and then perform a connected-components labelling. Of course, the choice of t has some effect on the outcome. A smaller threshold may produce many false detections whereas a larger threshold, although less prone to false detections, may miss some of the objects of interest. We explore the effect of t further in section 4. Figure 3 (right) shows a result of this procedure performed on the motion image shown on the left, where every extracted object is highlighted by the bounding box of its corresponding component (or blob) in the motion image.

2.3. Passing objects to the tracker

The final step in the motion detector considers the extracted regions, those believed to correspond to individual moving objects in the current frame, and creates a list of objects that should be added to those that are already being tracked. Each detected object is placed in one of the following three categories:

1. an object already being tracked;
2. an object not yet tracked but under consideration;
3. an entirely new object.

Objects are placed in the first category if they are found to be in close proximity to objects from the previous frame that are already being tracked. They are excluded from the list of new objects to be passed to the tracker.

There is also a "possible object" queue that contains objects currently under consideration for tracking (those that fall into category 2 above). An object has to appear consistently in several frames (say 5 or so) before it is passed to the tracker. The motivation behind this is simply to avoid passing false detections to the tracker. If the position of an object in the current frame is found to be close to an object already in the queue, it falls into category 2 above, and the corresponding element in the queue is updated. Because objects need to remain in the queue for a number of frames before being passed to the tracker, those that are not updated at the current frame are discarded. New objects (category 3) are simply added to the queue.

Objects that remain in the queue long enough are passed to the tracker, to be included among those that are already being followed.

3. Tracking

The tracking of moving objects is performed by a particle filter. We adopt a hierarchical approach [10], where objects are represented by several different descriptors. The tracker begins by calculating fast first-stage descriptors that are coarse and may yield many false positives. Good matches are then passed to a second stage where slower, more robust descriptors are calculated. Such a hierarchical approach allows for considerable speed-ups in the execution of the particle filter.

3.1. The particle filter

The particle filter can be seen as a non-linear generalization of the Kalman filter. It also allows for the noise models to be non-Gaussian. Here we provide a brief overview of how we utilize the particle filter. For a more in-depth discussion, including a derivation, the reader is referred to [12].

Basically, the "particles" in the particle filter represent possible solutions to the tracking problem. In our case each particle would represent possible (x, y)-coordinates for the object being tracked. Particles are adjusted iteratively according to the model equations governing the motion of the objects being tracked. After this adjustment each particle i is compared to some expected model (or template) and assigned a distance value d_i, such that $d_i = 0$ would indicate an exact match and a greater value of d_i would imply lesser accuracy.

Every particle is given a weight according to its distance value, as follows:

$$w_i = \exp(-d_i^2/(2\sigma^2)). \tag{5}$$

The weights of all the particles are then normalized so that they sum to 1. Note the manner in which the value of σ scales the distances. A large σ means that the values of $d_i^2/(2\sigma^2)$ are closer together. The weights of the particles would then change slowly because the weights of poor matches do not differ greatly from those of good matches. Hence the filter places more emphasis on the model equations than on the particle matches. The opposite holds for a small σ value.

There are two popular choices when it comes to the output of the particle filter, which should be the "best" estimate of the object's current position. One is to return simply the particle with the largest weight. The other is a weighted average of all the particles, where the state of each particle (in our case position coordinates) is weighed by its w_i value. We use this second option in our implementation.

An important issue worth mentioning is that of particle degradation. After several iterations the weights of many particles may drop down to zero or values very close to zero. In such a situation it is useful to perform particle re-sampling, allowing for more particles to contribute towards the tracking.

3.2. Descriptors

When using the particle filter to solve a problem one needs to decide on the nature of the particles. For the problem in this paper the particles are chosen to be coordinate pairs that represent possible locations of the object that we are tracking. Various descriptors are then calculated in a neighbourhood around the pixel represented by each particle.

The descriptors that are calculated for each particle are compared to some model value that corresponds to the expected value of the tracked object. After the final weights for all the particles have been calculated, the output of the filter is calculated as the weighted average. Model values for the various descriptors, that will be used in the next frame of the video sequence, are then calculated for this point.

3.2.1. First-stage descriptors

For first-stage descriptors in the hierarchical particle filter we again turn to the rectangle features of Viola and Jones [11], this time those depicted in Figure 1(b) and (c). These features are chosen because of the speed at which they can be computed. Further efficiency is possible here by computing them from integral images; see [11] for details.

Two rectangle feature descriptors are calculated for every particle. The absolute difference between those and the expected features of the model is calculated, and weights are obtained using (5). Every particle now has two weights, one for each rectangle feature. In order to determine a final first-stage weighting, the two weights can be multiplied together as this causes a high score for only those particles that have a high matching score in both cases (for both features).

3.2.2. Second-stage descriptors

Only those particles that have contributable weights from the first stage, i.e. those that have a first-stage weight greater than some threshold, are carried through to the second stage in the hierarchical filter. Here a histogram of oriented gradients [3] is computed as it is a more precise descriptor than rectangle features.

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: G_h that highlights horizontal edges and G_v that highlights vertical edges. The magnitude and angle of the gradient vector at each pixel is then calculated as

$$m(i,j) = \sqrt{G_h(i,j)^2 + G_v(i,j)^2}, \quad (6)$$

$$\alpha(i,j) = \arctan\left[G_v(i,j)/G_h(i,j)\right]. \quad (7)$$

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 distance 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) [13], 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 our purposes. Cha and Srihari [14] proposed a measure related to the EMD but much faster to calculate. Because gradient orientations range between 0° and 360°, with the endpoints regarded as equal, we use their modulo distance measure (as explained in full detail in [14]).

First- and second-stage weights are multiplied and the weights of all the particles are normalized to produce the final particle weights used for estimating the current position of every tracked object in the frame.

4. Experiments

In this section we examine the performance of the various components of the system by experimenting on a few video sequences of field hockey players. The motion detector is first evaluated as an individual system. We then investigate how well the motion detection stage performs in its primary role of passing objects to the tracking system. Finally the success of the tracker is considered.

4.1. Motion detection

The motion detection subsystem was applied to two video sequences of 200 frames each. Two quantities were measured: precision and recall. Precision is a measure of how accurate the detection is and is calculated as the number of correct detections (players) divided by the total number of detections. Recall gives an indication of how complete the detection is and gives the number of correct detections divided by the number of objects (in this case players) that are actually present. Of course, ideally these two values should both be as close as possible to 1. In an effort to measure precision and recall, manual annotation of the video frames were performed.

A crucial user-specifiable parameter affecting precision and recall in the motion detection system is the threshold t discussed in section 2.2. It specifies the amount of motion needed for a blob to be classified as a moving object. Table 1 below lists the obtained precision and recall for the two video sequences, for a few different values of the threshold t.

The poor performance of $t = 5$ in both sequences comes as a result of over-detection. Many regions that do not actually contain moving objects are falsely detected. An increase in t leads to fewer false detections but, at some stage, starts to exclude true objects from being detected (i.e. low recall).

Most of the missed and incorrect detections occur as a result of one player occluding another or when two players are close together. In the first case only the player visible to the camera is detected correctly. In the second case it may happen that neither of the players is detected correctly because they appear as one large blob. Figure 4 gives an example of each of these two problems. Missed detections can also result from players standing still or moving very slowly for a long period of time.

Sequence 1					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.60	0.85	0.94	0.97	0.99
Recall	0.77	0.87	0.84	0.52	0.33

Sequence 2					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.78	0.91	0.94	0.95	0.98
Recall	0.76	0.85	0.77	0.54	0.15

Table 1: *Precision and recall of the motion detection on the two video sequences. The threshold t specifies the amount of motion necessary in a region for it to be classified as a moving object.*

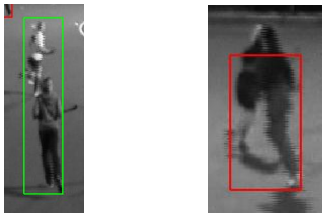

Figure 4: *Typical examples of situations causing faulty detections: one player occluding another (left) and two players in close proximity to each other (right).*

4.2. Passing detection results to the tracker

The experimentally obtained precision and recall of the motion detector, as a stand-alone system, are not bad but not exceptionally good either. However, as mentioned before, it is not that important that a player be detected at the very first possible instance.

Far more crucial is the motion detector's success rate at passing players to the tracker. Recall that an object needs to be detected for n frames before eventually being passed on. For these experiments we used $n = 5$. In order to investigate the success of the motion detection stage in the larger tracking system, we measure the following 4 quantities:

(i) average number of frames taken to pass a new object to the tracker;

(ii) number of true objects missed entirely and never passed to the tracker;

(iii) number of incorrect detections (non-players) that are passed to the tracker;

(iv) number of true objects correctly passed to the tracker.

These measurements are presented in Table 2 for the two test video sequences. Table 3 lists for each of the players visible in the two sequences the number of frames that she is in view before being passed on to the tracker by the motion detector.

In the first sequence there are no missed or incorrect handovers. The average time taken by the motion detector to hand over players is rather slow but, as is apparent from Table 3, this is mainly due to the two players. The first of these (a_4) entered the scene in close proximity to another player and only once she moved away from the other player was she picked up by the motion detector as a separate object. The second player with a long detection time (a_5) was the goalkeeper, who stood still for a long time, and was only detected after significant movement.

The missed detection in the second sequence resulted from a player that entered the scene in close proximity to another player, towards the end of the video, and never moved far enough away from the other player for her to be detected as a new player. The incorrect detection in the second sequence

	Sequence 1	Sequence 2
(i) avr time per handover	27.3	14.8
(ii) missed players	0	1
(iii) incorrect handovers	0	1
(iv) correct handovers	6	11

Table 2: *Measurements indicating the success of the motion detection system at its primary role of handing over correct objects to the tracker.*

Sequence 1						
player	a_1	a_2	a_3	a_4	a_5	a_6
detection time	5	8	19	35	88	9

Sequence 2										
b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
5	5	5	5	5	5	5	5	8	15	100

Table 3: *Detection times for all the players in the two video sequences. Each is given as the number of frames that the player is in view before being passed to the tracker by the motion detector.*

came as a result of slight camera movement so that the background model was, for a short period of time, inaccurate. Player b_{11} also entered the scene in close proximity to another player, hence the longer detection time.

4.3. Tracking

Without an independent set of tracking results to measure our tracking system against it is difficult to perform a meaningful evaluation. As such we provide here some examples of how the tracker functions in a number of different situations: a "normal" sequence of frames, a sequence containing a partial occlusion, and one that contains a full occlusion.

Figure 5 depicts the tracking results for a sequence of 100 frames of two players. The last frame in this sequence is shown, and the small yellow dots indicate the paths followed by each of the tracker region centers. In this sequence the tracker successfully follows the targets.

A partial occlusion (from the camera's point of view) occurs when one player covers part of another for some period of time. If the area of occlusion is small so that most of the partially occluded player remains visible, the tracker is able to follow her correctly. Figure 6 shows an example before, during and after such a partial occlusion.

In a full occlusion one players obscures another more-or-less completely. Figure 7 shows an example. In this case the tracker loses the occluded player and may, after the occlusion when the two players separate, erroneously follow the occluding player. This may happen because (in our current system) template models are updated rather quickly so that, in the event of a full occlusion, both trackers may lock onto the front-most player.

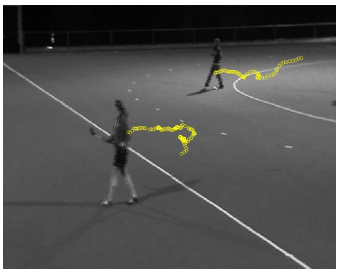

Figure 5: *Depiction of tracking results on a sequence of 100 frames. The dots indicate estimated positions of the players for every frame.*

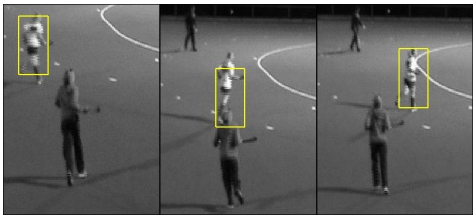

Figure 6: *Before, during and after a partial occlusion. The tracker successfully follows the partially occluded player.*

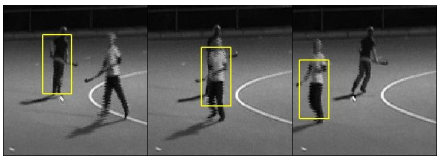

Figure 7: *Before, during and after a full occlusion. The tracker on the occluded player locks onto the wrong player after the occlusion due to incorrect model updating.*

4.4. Speed performance

The various components of the system were designed with speed of execution as a high priority. We implemented a proof-of-concept in Matlab (which is notoriously slow). The motion detection runs at about 4 frames per second, which does not depend significantly on the number of players in view. The speed of the tracker is more dependent on the number of object being tracked, as separate filters run on each. The table below lists the speed, in number of frames per second (FPS), for various numbers of tracked players.

number of players	1	2	3	4	5
FPS for tracking	6.5	3.4	2.4	1.7	1.4

Apart from executing compiled code (such as C++), further speed increases would be possible from the parallelization of various parts. The entire calculation of the motion image is a prime candidate.

5. Conclusions and future work

In this paper we considered the problem of locating and tracking sports players in a grayscale video sequence. We developed a reliable and fundamentally fast motion detector that passes new players entering the view to a tracker. A particle filter performs the tracking, and is optimized for speed by a hierarchical approach with fast descriptors.

Experiments show that the motion detector, despite its simplicity, maintains high levels of accuracy. Missed or incorrect detection do occur from time to time, but the detector succeeds well in its primary role of eventually finding players and disregarding false detections before passing them to the tracking system. Currently the tracker functions well when players maintain a reasonable distance from one another, but may deteriorate slightly in the presence of occlusions.

Future work may include a more integrated combination of the motion detection results with those of the tracker, for more precise tracking. This integration may also prove useful in detecting and handling occlusions, so that the filter tracking an occluded player can be reset after the occurrence of the occlusion. Some form of player recognition, based for example on face or digit recognition, can also be implemented to boost the tracker's confidence in telling players apart after an occlusion.

6. Acknowledgments

We thank the National Research Foundation for financial assistance. We also thank McElory Hoffmann and Ben Herbst for insightful discussions regarding the particle filter.

7. References

[1] H. Schneiderman, T. Kanade, "A statistical method for 3D object detection applied to faces and cars", IEEE Computer Vision and Pattern Recognition, 1746–1759, 2000.

[2] D. G. Lowe, "Object recognition from local scale-invariant features", International Conference on Computer Vision, 2:1150–1157, 1999.

[3] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection", IEEE Computer Vision and Pattern Recognition, 2:886–893, 2005.

[4] K. Mikolajczyk, C. Schmid, A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors", European Conference on Computer Vision, 1:69–81, 2004.

[5] T. Horprasert, D. Harwood, L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection", ICCV Frame Rate Workshop, 1–19, 1999.

[6] C. Stauffer, W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", IEEE Computer Vision and Pattern Recognition, 2:2246–2252, 1999.

[7] A. Elgammal, D. Harwood, L. Davis, "Non-parametric model for background subtraction", European Conference on Computer Vision, 2:751-767, 2000.

[8] P. F. Gabriel, J. G. Verly, J. H. Piater, A. Genon, "The state of the art in multiple object tracking under occlusion in video sequences", Advanced Concepts for Intelligent Vision Systems, 166–173, 2003.

[9] X. Song, J. Cui, H. Zha, H. Zhao, "Probabilistic detection-based particle filter for multi-target tracking", British Machine Vision Conference, 223–232, 2008.

[10] C. Yang, R. Duraiswami, L. Davis, "Fast multiple object tracking via a hierarchical particle filter", International Conference on Computer Vision, 1:212–219, 2005.

[11] P. Viola, M. J. Jones, "Rapid object detection using a boosted cascade of simple features", IEEE Computer Vision and Pattern Recognition, 1:511–518, 2001.

[12] M. Hoffmann, K. Hunter, B. Herbst, "The hitchhiker's guide to the particle filter", 19th Sympsium of the Pattern Recognition Association of South Africa, 33–38, 2008.

[13] Y. Rubner, C. Tomasi, L. J. Guibas, "A metric for distributions with applications to image databases", International Conference on Computer Vision, 59–66, 1998.

[14] S.-H. Cha, S. N. Srihari, "On measuring the distance between histograms", Pattern Recognition, 35:1355–1370, 2002.