Research Article

A Reliable Event-Driven Strategy for Real-Time Multiple Object Tracking Using Static Cameras

Amany M. Sarhan, Ahmed I. Saleh, and Ramy K. Elsadek

Department of Computer, Faculty of Engineering, Mansoura University, Mansoura, Egypt

Correspondence should be addressed to Ahmed I. Saleh, aisaleh@yahoo.com

Received 24 November 2010; Revised 5 March 2011; Accepted 20 April 2011

Academic Editor: Deepu Rajan

Copyright © 2011 Amany M. Sarhan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, because of its importance in computer vision and surveillance systems, object tracking has progressed rapidly over the last two decades. Researches on such systems still face several theoretical and technical problems that badly impact not only the accuracy of position measurements but also the continuity of tracking. In this paper, a novel strategy for tracking multiple objects using static cameras is introduced, which can be used to grant a cheap, easy installation and robust tracking system. The proposed tracking strategy is based on scenes captured by a number of static video cameras. Each camera is attached to a workstation that analyzes its stream. All workstations are connected directly to the tracking server, which harmonizes the system, collects the data, and creates the output spatial-tempo database. Our contribution comes in two issues. The first is to present a new methodology for transforming the image coordinates of an object to its real coordinates. The second is to offer a flexible event-based object tracking strategy. The proposed tracking strategy has been tested over a CAD of soccer game environment. Preliminary experimental results show the robust performance of the proposed tracking strategy.

1. Introduction

Because of the advance in surveillance systems, object tracking has been an active research topic in the computer vision community over the last two decades as it is an essential prerequisite for analyzing and understanding video data. However, tracking an object, in general, is a challenging problem. Difficulties in tracking objects may arise due to several reasons such as abrupt object motion, changing appearance patterns of the object and/or the scene, nonrigid object structures, partial/full object-to-object and object-to-scene occlusions, camera motion, loss of information caused by projection of the 3D world on a 2D image, noise in images, complex object motion, complex object shapes, and real-time processing requirements. Moreover, tracking is usually performed in the context of higher-level applications, which in turn require the location and/or shape of the object in every captured frame. Accordingly, several assumptions should be considered to constrain the tracking problem for a particular application.

A great deal of interest in the field of object tracking has been generated due to (i) the recent evolution of high-speed computers, (ii) the availability of high quality and inexpensive sensors (video cameras), and (iii) the increasing demand for an automated real-time video analysis. Tracking an object is to get its spatial-tempo information by estimating its trajectory in the image plane as it moves around a scene, which in turn helps to study and predict its future behavior. There are three main steps in video analysis: detection of interesting moving objects, tracking those objects from frame to frame, and analysis of the recognized object tracks to detect their behavior. Some examples for tracking include the tracking of rockets in a military defense system, customers in a market, players in a sport game, and cars in a street. Tracking these objects helps in military defense, goods arranging, sport strategies, and traffic control, respectively.

Basically, tracking can be done by one or multiple types of sensors [1]. Radar, sonar, infrared, laser radar, and video cameras are the common used tracking sensors.

FIGURE 1: The three scenarios of surveillance system.

Moreover, based on the resolution of the input sensors and the preciseness of the analysis, the surveillance system may work in one of three scenarios [2]. The first is the high preciseness scenario, which can be used in gait recognition to get the motion pattern of an object [3]. The second is the medium preciseness one, which can be used in recognizing generic activities such as walk, run, or slide [4, 5]. The third is the low-preciseness scenario, which is used to detect the presence of the object in crowded environment. Figure 1 gives an illustration of the three scenarios resolution.

Using color information may solve some tracking situations. Various methods have been proposed to present the appearance model of the tracked objects, such as, polar representation method [6]. Another similar method has been developed by slicing the object and taking the average color in each slice. These colors were stored as the appearance template of the tracked object. Figure 2 illustrates the appearance model of a soccer player.

Sometimes, it may be more beneficial to replace the high-preciseness sensor by multiple low-preciseness ones. Such strategy has a great impact in cost minimization as the low-preciseness sensors are cheaper than those high-preciseness ones [7–9]. Also, it improves the tracking preciseness by dividing the wide tracking area into several smaller divisions [10]. Moreover, this strategy guarantees reliable object recognition and solves the problem of object detection in the cluttered environments. Those benefits are gained as the multiple-sensors tracking system provides more information that can be used for further analysis and prediction. For instance, the 3D position of a soccer ball can be predicted using a couple of 2D frame camera-video streams.

However, multiple-sensors tracking systems suffer from several hurdles, such as sensors registration and integration. Furthermore, they become very complicated if the design of sensors network is noncentralized. In the centralized network scenario, the data streams from the machine

FIGURE 2: The appearance template of a tracked object.

behind the sensor (workstation) to a dedicated server, and vice-versa. So, the data stream paths are clearer than those in noncentralized systems. Generally, the centralized networks (server-workstation network) are preferred than the noncentralized networks (peer-to-peer) as they usually introduce better stability, robustness, and organization.

Usually, in several tracking systems, the tracked objects are not only moving randomly (not as rigid bodies) but also captured in low resolution. These restrictions result in several hurdles that are usually faced during the tracking systems' operation. Such hurdles include (i) the automatic initialization and installation of new objects, (ii) detection of objects that enter the field-of-view (FOV) of each camera, (iii) detection of those exiting objects, (iv) guaranteeing

an objects' continuous migration from a camera's FOV to another's, and more importantly, (v) solving the merging/splitting situations of objects or groups of objects, which are the major challenge in the tracking system.

In this paper, a real-time tracking system will be proposed for automatically tracking a set of moving objects using a number of video streams collected by a set of static cameras. The simple installations, portability, as well as the low cost are the salient features of the proposed tracking system [11]. The system uses multisensors (static cameras), which are constructed using a centralized network, to track several moving objects. Such infrastructure has many benefits such as (i) maximizing the object tracking accuracy, (ii) minimizing not only the processing overhead but also the ambiguity in objects occlusion, (iii) enhancing both the systems portability and installation flexibility, and (iv) increasing the capability of event recognition and video parsing [12–14].

The main contributions of this paper are (i) introducing a novel image-to-world coordinates transformation that impacts well the systems' setup and tracking phases and improves the flexibility of tracking those objects that travel between cameras FOV(s), (ii) a new concept that considers the moving object as a group of physical targets, which in turn solves and corrects a lot of tracking problems, has been developed, and finally, (iii) introducing a novel event-driven tracking algorithm for efficient tracking of multiple objects using multiple cameras network.

The rest of the paper is organized as follows. Section 2 gives a brief review of the previous efforts. Section 3 presents some main concepts of object tracking systems. Section 4 discusses the proposed system architecture, phases, and algorithms. The experiments and their results are described in Section 5. Finally, conclusions are illustrated in Section 6.

2. Previous Efforts

Over the last two decades, a lot of work in the area of object tracking has been developed. Both the structure and organization of those developed systems differ according to their different purposes, the physical properties of the used sensors, and the considered economic issues. To the best of our knowledge, based on the underlying architecture, object tracking systems can be classified into three main categories. Two of them are built over a network of camera stations; the first depends on static cameras, while the second uses dynamic cameras in zooming and rotating. On the other hand, the third category depends on a wide-screen image (panorama image), which can be collected by a number of static cameras in the same side of the tracking field. Through the rest of this section, the first category, which employs the static cameras, will be discussed in details because of its similarity to our system. Afterwards, the two other types will be briefly reviewed.

According to the first category, the tracking system consists of a set of workstations that cover the tracked environment using a set of static cameras. Kang et al. [6] used multiple stationary cameras to provide a good coverage of a soccer field. The registration of cameras viewport was performed using a homography. Accordingly, the system can

integrate 3D motion tracking component without explicit recovery of the 3D information. The 3D estimation used the projective space instead of the Euclidean, as the system does not use the camera calibration parameters. However, such system posed several issues that need to be addressed such as reducing the complexity and the delay generated by the propagation of uncertain hypothesis. Another system had been introduced by Xu [12, 15], which used eight digital video cameras statically positioned around the stadium, and calibrated to a common ground-plane coordinate system using Tsai's algorithm [16]. Based on Ming's work, a two-stage processing architecture had been used. The first was the extraction of information related to the objects observed by each camera (from the input image streams). Then, the data from each camera was entered to a central tracking module to update the estimated state of the tracked objects. However, there are several difficulties and opportunities to be addressed in the system reliability and accuracy, such as the movement of a group of occluded objects from a workstation's FOV to another.

Another architecture that uses stationary cameras was proposed by Iwase and Saito in [17, 18]. The system uses multiple view images to avoid the occlusion problem, so that it can simply obtain a robust object tracking. Initially, an inner-camera operation is performed independently in each camera to track the players. When an object cannot be tracked, an intercamera operation is performed as a second step. Tracking information of all cameras is integrated by employing the geometrical relationship between cameras through a process called homography. Inter-camera operation makes it possible to get the location of the object that is not detected in the input frame. Such an object is the one that is completely occluded by another ones; hence, it becomes outside the angle of view. However, in spite of its salient properties, the system suffered from the difficulty of matching the object-scene positions between the camera's input frames. Moreover, unclear data streams and complex manipulation were the main hurdles that had a harmful impact on the performance of such tracking system.

On the other hand, the tracking systems architectures that use a network of dynamic cameras are sometimes used as a result of their cheapness. For illustration, tracking in soccer game needs no further cost as it depends on a number of dynamic cameras to deliver TV-Camera video streams, which is already constructed. Hayet presented a modular architecture for tracking objects in the sports broadcasting [19]. However, the system faced a lot of complexity, uncertainty, and inaccuracy. Furthermore, as a result of complex calculations, the system suffered from the long processing time, which results in the inability to process on the real time. On the other hand, its practical implementation depends frequently on manual feeds.

Finally, the use of a wide image (panorama image) makes it possible to track objects using one machine, which avoids the difficulties of correlating data over multiple machines. Sullivan and Carlsson in [20] used such architecture to track objects in soccer game. However, his system required specific, expensive, and calibrated sensors. Furthermore, the system consumed a long time to process such a large video frame.

FIGURE 3: Capturing interesting moving objects using background subtraction method.

3. Background and Basic Concepts

Through the next subsections, a quick review of the methodology used to extract the tracked objects as well as the employment of Kalman filter in the literature will be introduced.

3.1. Extracting Tracked Objects. To extract the moving objects in the video stream a very faster approach is the background subtraction method [21], which compares each frame against the background model. Background subtraction delineates the foreground from background in the images. Initially, it estimates the background from a certain number of early frames without any object inside. Subsequently, the following image frames are compared to the estimated background and calculates the difference in pixel's intensity. The drastic changes in pixel's intensity indicate that the pixel is a motion pixel. In other words, the different pixels, from those in the background model, in the current frame present the foreground pixels which, in turn, may represent the interesting moving objects. The background subtraction method is illustrated in Figure 3.

Definition 1. Background Model $BM_t(x, y)$. The Background model $BM_t(x, y)$ is an image which presents, at each spatial location (x, y) , the pixels color or luminosity of static objects in camera's FOV such as ground and walls. It can be updated regularly along time t based upon its creating technique.

Definition 2. Foreground pixel $FG_t(x, y)$. Foreground pixel $FG_t(x, y)$ is an image which different from its equivalent in background model, at spatial location (x, y) and time t , in color or luminosity by specified threshold T_s . So, for any frame $F_t(x, y)$,

$$FG_t(x, y) = 1 \quad \text{if } |F_t(x, y) - BM_t(x, y)| > T_s. \quad (1)$$

Two different techniques can be used for creating the background model: recursive [22] and nonrecursive [23]. A nonrecursive technique stores a set of previous frames into a buffer and then builds the background model based on the variation of each pixel within those consecutive frames. On

the other hand, the recursive technique does not use a buffer. It builds the background model by recursively updating it with each input frame. The recursive technique requires less storage rather than the nonrecursive one; however, any error done in an input frame will have a harmful effect on the model. After extracting the foreground pixels by a segmentation process (such as background subtraction), a filter (such as Laplacian of Gradient (LoG)) is applied. LoG is used to reduce not only the noise in foreground pixels, but also the noise in the edges of the pixels groups.

Next, the objects are created by connecting foreground pixels. For each object, a number of foreground pixels are clustered as a blob. These pixels are clustered by applying Otsu's method, which determines the threshold for each pixel either belongs to an object or not. This blob is the source of object's appearance in terms of its dimensions, its color signature, and even its position. The object may be represented by more than one blob. Finally, the object dimensions and position are represented with a bounded rectangle around its blobs. Figure 4 illustrates these steps in simple manner.

Definition 3. Tracked Object. O_m A tracked Object O_m can be represented by a bounding rectangle which groups a number of closest blobs. For the blobs included in N pixels $p(x, y)$, the position of the tracked object $O_m(x_m, y_m)$ is the center lower pixel of bounding rectangle, as shown in the following formula:

$$x_m = \sum_{i=1}^N \frac{p_x}{N}, \quad y_m = \sum_{i=1}^N \frac{p_y}{N}. \quad (2)$$

The bounding rectangle of tracked object O_m has its width w and height h expressed by the following formula:

$$w_m = \left\lceil \max_{i=1}^N p_x - \min_{i=1}^N p_x \right\rceil, \quad h_m = \left\lceil \max_{i=1}^N p_y - \min_{i=1}^N p_y \right\rceil. \quad (3)$$

The tracked object is always expressed by $O_m = \{x_m, y_m, w_m, h_m\}$.

Step	Description
a	Input frame
b	Extracted foreground pixels
c	Clustered blobs
d	A rectangle represented tracked object

FIGURE 4: Steps of creating the object from by background subtraction method.

3.2. Tracking Objects Using Extended Kalman Filter. An object tracking system is a system that determines the location of an object(s) over time using a sequence of images (frames). So, the first idea for getting the location of an object in the next image is to search for it in a small rectangle area around its present location. However, such idea may fail if the object got out of the searching rectangle. Hence, it is needed to modify this method to predict the object location in the next image by searching for it in a small rectangle area. To accomplish such aim, Kalman filter can be successfully employed because of its ability of prediction in noisy environments [24]. However, the main problem faced when using Kalman filter in real-time tracking systems is that it requires a statistical model not only for system output data but also for its measurement instruments. This problem can be avoided by using a recursive algorithm, as illustrated in Figure 5, which adjusts these unknown parameters, such as the measurement noise variance after each time step based on the observed measurements.

Kalman filter works as an estimator for the process state at some time and then obtains feedback in the form of noisy measurements. So, the equations of Kalman filter are divided into two groups: the prediction (time update) and correction (measurement update). The solved process by Kalman filter is governed by a linear stochastic difference equation. In the tracking systems, this process is nonlinear, so Kalman filter must be modified to tackle this problem. The modification of the filter comes in a famous form called extended Kalman filter (EKF) [24].

4. The Proposed Strategy

Through this section, the proposed tracking strategy will be illustrated in details as well as its main components.

4.1. Preliminary. The proposed tracking system architecture, as well as its main elements, is illustrated in Figure 6. The system consists of a flexible number of workstations, each analyzes the video stream coming from a static camera.

Those cameras are the system sensors, which cover the overall considered tracking field. The system is centralized at the server machine, which collects, integrates, and corrects data coming from workstations and then sends it to a media server such as web servers, multimedia servers, TV shows, or data-mining servers.

As illustrated in Figure 6, our proposed tracking system can be partitioned into four main layers, which are (i) cameras layer, (ii) workstations layer, (iii) central server layer, and (iv) data representation layer. The cameras layer contains a number of static cameras that deliver the input video streams. Each camera covers a specific region in the field under scope which is called the camera's FOV. Furthermore, the cameras FOVs must partially intersect especially in Exit/Entry boundaries and preferably in crowded spots (such as the penalty area of soccer pitch). Ideally, the intersecting scope cameras should be positioned orthogonally, along their projective axis over the pitch as shown in Figure 7, or near orthogonally. Such construction solves most occlusion problems and can be used to get the 3D position of some objects.

The overall system's FOV encompasses the visible FOVs of all the cameras in the system. On the other hand, the system is not interested in any virtual FOV (the uncovered areas by cameras). Each camera is connected to a workstation as illustrated in Figure 6. Each workstation reflects its camera position in the system architecture and works as the local tracker of video stream coming from such camera. Hence, it can create its local database that expresses the spatial-tempo data of those objects in its camera's FOV. Furthermore, all workstations are connected to an ETHERNET LAN, which connects them to the tracking server.

In the server layer, the server machine works as the system's central point. It receives the spatial-tempo data from each workstation and then collects them to (i) guarantee the data integration, (ii) recorrect falsely received data, and (iii) predict the most correct data. Hence, the data streams from workstations to server, and vice versa. Finally, the output spatial-tempo data from the system is then stored

FIGURE 5: Kalman filter recursive algorithm flow chart.

FIGURE 6: System architecture layers and elements.

FIGURE 7: The ideal positioning of a couple of cameras. The two cameras' axes are perpendicular in top view.

FIGURE 8: Workstation tracking methodology.

in a database. This database can be expressed by general markup language like XML to be easily sent at real time to another media over a low bandwidth, TV shows, mobile cells, or internet. In addition, the complete database can be linked to video streams and can be used offline as a multimedia database to predict situations happened in the tracked environment as a kind of data mining.

Before tracking, each workstation's FOV must be defined with respect to the general system coordinates to avoid gaps and blind areas in the tracked environment. The workstation's FOV is defined as a polygon that has a number of corners representing the pitch area covered by the camera in front of that workstation. The manual choosing of these corners is more accurate, less expensive, and more transparent in many situations. Choosing those corners is performed over one captured frame before beginning tracking.

4.2. Workstation Operation. At each workstation, tracking moving objects is performed using the methodology illustrated in Figure 8. Segmenting static-camera video stream is handled by background-subtraction method as illustrated in Section 3.1. Background model can be created manually and updated frequently by a recursive technique such as approximated median filter [23]. After subtracting the input frame by the background model, the foreground pixels are often heavily fragmented, so a filter is applied to reduce the noise [25]. A morphological filter is then used to insure that every group of pixels makes up a single object. The moving objects are created by fast grouping foreground pixels as blobs. Then, each object is created by grouping closest blobs using a connected component analysis. Those blobs are surrounded by a bounding box, which presents the

tracked object. Tracking each object in the scene is done by EKF [26]. Furthermore, EKF is employed to solve occlusion problem which happens when two or more tracked objects merge together. The prediction stage in EKF model of the occluded object and the foreground position of the occluded group are incorporated to update the position of each object in the group [14].

After splitting the occluded group, workstation depends on three choices to match the input objects with the output objects of the occluded group, which are (i) EKF prediction and correction, (ii) the color histogram of each object, and (iii) the feedback coming from the server.

4.3. Server Operation. The server machine acts as the center of the system as it receives all data coming from the workstations connected to the cameras (as shown in Figure 6). However, the data stream does not flow only from the workstation to the server but the server may also send its feedbacks to workstations. These feedbacks may be sent when (i) the server corrects data about objects tracks over a workstation; (ii) the server solves the uncertain data about occluded objects, or (iii) the entry of an object to a workstation's FOV while exiting from another. Hence the server performs a set of critical tasks which are (i) collecting and synchronizing data from each workstation, (ii) associating data about each object received from each workstation, and (iii) fusing all data to get the global position of each target in real world coordinates. The servers' operation is depicted in Figure 9.

Data synchronization is performed by the scheme presented in [19]. The data sent from the workstations are time stamped. However, data coming from different workstations may arrive at server at different times. Thus, the server waits

FIGURE 9: Server jobs and interactivity with workstations.

a short period (which should be less than the rate in which workstations cameras capture frames) before processing all incoming data stamped with the same time. Any data arriving after this period will be discarded. Data coming to the server are processed by two modules which are; data association and data fusion modules. Data association is the technique for correlating observations from multiple sensors about the tracked object [13, 27]. Hence, data association module gives a unique label to every tracked object. Many algorithms can be employed to build this module such as Nearest-Neighbor Standard Filter (NNSF) [28], Optimal Bayesian Filter (OBF) [29], Multiple Hypothesis Tracking (MHT) [30], Probabilistic Data Association (PDA) [31], Joint PDA (JPDA) [32], Fuzzy Data Association (FDA) [33], and Viterbi Data Association (VDA) [34]. Although PDA is recommended in cluttered environment, we used JPDA algorithm because it extends PDA to simultaneously track several objects. Moreover, JPDA is considered to be the best compromise between performance and complexity. After every target is associated to itself in every sensor stream, the data from all streams must be fused to get the global position. Data fusion can be done by many models such as Intelligence Cycle, Joint Directors of Laboratories (JDL) Model [35], Waterfall Model [36], and Omnibus Model [37]. Omnibus Model is a recent model which capitalizes the older models advantages and minimizes their limitations. Hence, the explicit structure of model components makes Omnibus Model the preferable choice in our data fusion module.

4.4. Image-to-World Coordinates Transformation. In any tracking system, transforming the image-to-world coordinates is a very essential and complicated step. This transformation should be calculated accurately to get the precise real position of each moving object. However, depending

basically on the internal parameters of the camera and its position makes this transformation very complicated and expensive. Camera calibration is one of these complicated ways which is proposed in [28]. However, this method is based on the calibration camera parameters using costly hardware and usually based on complicated calculations that may not be possible in large environments. As we aim to establish a simple, cheap, and real-time system, we avoid the complicated transformation methods. However, our work focuses on tracking objects moving on flat planes such as pitch or parking areas; hence, we will present a novel method for transforming the 2-dimensional ground coordinates in the image to real-world coordinates. Such transformation is employed at each workstation. The proposed transformation method is based on the manual assignment of a predefined group of points that their real-positions are known. Those points are arranged in perpendicular lines to illustrate a grid, which is called the transformation grid (TG) as defined in Definition 4.

Definition 4. Transformation Grid (TG). Transformation Grid (TG) is a set of points which are arranged in perpendicular rows and columns over the transformed plane. These points' coordinates are known in both real and image coordinates. For easy and explicit calculations, the distances between each point and any neighboring point are the same in all directions.

At the server, a TG-creator module creates the TG for each workstation. Initially, TG-creator reads the real coordinates of the TG points' real coordinates which are organized and measured manually. These data are presented in the form of an XML file such as that is shown in Algorithm 1.


```

<?xml version = "1.0" encoding = "utf-8"? >
<scene>
  <point name = "... " x = "... " y = "..."/>
  <point name = "... " x = "... " y = "..."/>
  .....
  .....
</scene>

```

ALGORITHM 1

The second step in the TG-creator is to use the graphical user interface (GUI) shown in Figure 10, which employed to manually assign points in real model to its equivalent in the image. To accomplish such aim, TG-creator plots the real model points, as illustrated in the interface shown in Figure 10(b), by reading the previous XML file. Then, the server's operator chooses an initial frame of any workstation. This frame is captured with presence of TG points only. For instance, the human operator, as shown in Figure 10(a), chooses the initial frame of middle workstation over the pitch and begins to assign some points on both sides.

Hence, by simple clicks using TG-creator interface, the operator can assign each point in the initial frame, at the left part, to its equivalent, at the right, in the real model. As soon as the manual assignment is finished, the last step in TG-creator is to produce an XML file that collects the coordinates of each image point $P_{mi}(x_{mi}, y_{mi})$, got by the workstation's initial frame, and its equivalent in real coordinates $P_{mr}(x_{mr}, y_{mr})$. So, any point P_m in the TG is expressed as $P_m = \{x_{mr}, y_{mr}, x_{mi}, y_{mi}\}$. Furthermore, TG points are grouped into TG cells as defined in Definition 5. Also, the points which belong to incomplete cell in the workstation initial frame are ignored.

Definition 5. TG-Cell. For any TG having points arranged by their real coordinates in rows j and columns k , the TG-cell is expressed as $Cell_i = \{P_{j,k}, P_{j+1,k}, P_{j,k+1}, P_{j+1,k+1}\}$. So, the TG-cell is a set of 4 closest points which present the four corners of a rectangle.

The XML file written by the TG-creator and expresses one workstation's TG is written as in Algorithm 2.

Transforming a point from its image coordinates to real coordinates is begins with reading the previous XML. Next, the TG-cell where the transformed point exists is detected, or the nearest TG-cell if it does not exist inside any one. Once the TG-cell is detected, the transforming algorithm can be started as illustrated in Algorithm 3.

Referring to Figure 11, a point P is found in cell $C = \{P_{11}, P_{21}, P_{22}, P_{12}\}$, and it is also laying on two lines which are $P_{x1}P_{x2}$ and $P_{y1}P_{y2}$, each of them is parallel to two edges and perpendicular to the other two edges. Depending on the fact that cross-ratios remain invariant under perspective projection, and using an approximation in this ratio as in [19]: the following pair of equations can be predicted

$$R = P_{11}P_{x1} : P_{11}P_{12}, \quad R = P_{21}P_{x2} : P_{21}P_{22}. \quad (4)$$

(a)

(b)

FIGURE 10: A part of TG-Creator interface. Numbers and black rectangles refer to some of equivalent points in both an initial frame in the left and real model in the right.

Hence, using the previous pair of equations, the real x -coordinate of point P can be calculated by iterating R . Referring to Figure 12(a), x -threshold is the perpendicular distance from point P to the line $P_{x1}P_{x2}$. By changing R , line $P_{x1}P_{x2}$ moves and x -threshold changes too. To reduce x -threshold under one pixel, R is iterated by the algorithm illustrated in Algorithm 3.

In the same way, the real y -coordinate of point P can be calculated using the algorithm illustrated in Algorithm 3 by considering the y -coordinates rather than the x -coordinates.

As soon as the image coordinates are transformed to the corresponding real coordinates, the workstations coverage modules can be started immediately. After constructing the TG, the workstations coverage modules help in plotting the coverage polygonal over the ground of each camera's FOV. This plotting is performed manually with a developed GUI software aid tool at the server. The server receives the initial frame of each workstation and its GT XML file, and then the server operator assigns corner points of the coverage polygonal manually. Figure 13 shows the GUI of the workstations coverage aid tool. It shows the initial frame with the corresponding plotting coverage polygonal of each workstation as well as the coverage polygonal of all workstations. Our workstations coverage aid tool guarantees the complete coverage of the area under consideration as well as the overlapping between cameras' FOVs. Hence, it helps the server operator to discover the blind areas, if they are found.

```

<?xml version = "1.0" encoding = "utf-8"? >
<workstation id = "...">
  <cell id = "...">
    <point name = "... " xr = "... " yr = "... " xi = "... " yi = "..."/>
    <point name = "... " xr = "... " yr = "... " xi = "... " yi = "..."/>
    <point name = "... " xr = "... " yr = "... " xi = "... " yi = "..."/>
    <point name = "... " xr = "... " yr = "... " xi = "... " yi = "..."/>
  </cell>
  .....
  .....
</workstation>

```

ALGORITHM 2

FIGURE 11: The considered point P with its hosting cell C .

4.5. Constraining the Environment and Tracked Objects. The main goal in any tracking system is to solve the problem of discontinuity of objects paths (tracks) during the tracking process. This problem occurs due to many reasons such as (i) the occlusion with objects or structures in the scene (disappearing partially or completely by a static object), (ii) object disappearing in blind areas, or (iii) exiting and reentering of objects from the scene. Basically, object occlusion can be categorized into two main categories which are (i) interobject occlusion and (ii) occlusion of objects. The former happens when a group of objects appeared in the scene where objects are blocked by others in the group. This poses a difficulty not only in identifying those objects in the group, but also in determining the exact position of each individual object. On the other hand, occlusion of objects occurs when the object is not observed or disappeared for an amount of time blocked by structures or objects (static or dynamic) in the scene. Hence, the tracking system has to wait for the object's reappearance or concludes that the object has left the scene. Hence, to minimize the effect of occlusion, we suppose an environment totally covered by several cameras; moreover, the ground is always visible since there are no static objects. On the other hand, the discontinuity of objects tracks has been overcome by (i) initializing new objects, (ii) successfully solving merging/splitting situations, and (iii) precisely integrating entering/exiting FOVs of the tracking workstations.

In most tracking systems, each physical target is represented as one corresponding object. However, this representation is not suitable for tracking multiple continual interacting targets. A generic formation of the tracked objects, in our system, assumes that "the tracked object can be represented as a group of micro-objects." This assumption is illustrated in Definition 6.

Definition 6. GOO Representation. A group-of-objects (GOOs) representation maps the tracked object to a set of physical targets.

Considering Definition 6, the ambiguous tracking cases, especially merge/split cases, will be explicitly solved. Also, the mystery of splitting object, after it has been registered over the system as one object, can be easily solved.

Figure 14 shows the situation in which three players are tracked at the first frame as one object. After some frames, one player splits and becomes a single object and the same for the second and third players. If the tracking module considers the three players at the beginning as a single object, it has to generate a new tracked object at each split. Also, the past trajectory of the split player will be lost. However, if the tracked players are considered as a group, the past trajectory of the split object would be the same as the previous trajectory of the group from which it has

FIGURE 12: A sample illustration of transformation algorithm basics and steps.

disjoined. By the same way, the trajectory of a group is the future trajectory of its merged objects or groups. So, GOO representation of the tracked objects could be useful in merge and split situations.

4.6. Object Tracking Strategy. In the traditional tracking system, the workstations are entrusted with segmentation and objects creating stage and objects paths creating stage. However, the server was entrusted with feedback correction stage. We break the linearity of cascading the classic tracking stages by “taking the decision of tracking from events performed in those stages.” An early version of this strategy was introduced in [8, 9]. The strategy of using tracking events promotes the tracking system automation. It allows the automatic object creation and avoids the manual scenarios. Using tracking events also maximizes the stability of objects traveling between system cameras. It also enhances the continuity of object tracking in several scenarios such as merging and splitting. The considered events are illustrated in Table 1.

TABLE 1: Tracking events used, their meanings, and their responses.

Event	Description	Corresponding response
E_1	Creating a new object	R_1
E_2	Object entering the FOV of workstation	R_2
E_3	Object split from a group of targets	R_3
E_4	Object exiting the FOV of workstation	R_4
E_5	Object merged with other one	R_5
E_6	Object is tracked in stable manner	R_6

Each event is triggered by a collection of conditions that are generated by simple operations, calculation, and conditions. For example, changing the size of object-bounded box R_{seg} that is generated by background subtraction, changing

Input data:
 The image coordinates of point $P(x_i, y_i)$.
 The TG-Cell point P laid in $Cell^i = \{P_{11}, P_{21}, P_{22}, P_{12}\}$.
 The real x-coordinate of point P_{11} is called P_{11-xr} .

Calculated parameters:
 R_n : the iterated value of R after n tries.
 R_0 : the initial value of R which equals 0.5.
 P_{x1}, P_{x2} : are calculated with equation (4) and by $Cell^i$ points' image coordinates.
 x_{th} : the perpendicular distance in pixels from point $P(x_i, y_i)$ to line P_{x1}, P_{x2} .
 P_{xr} : the real x-coordinate of Point P ; it is the goal of this algorithm.

Algorithm:
 Start:
 for $n = 0$; $R_0 = 0.5$;
 if x_{th} for $R_0 \leq 1$ pixel then $R = R_0$, and go to OUT
 Calculate R_n :
 $n = n + 1$
 $R_{n1} = R_{n-1} + |R_{n-1} - R_{n-2}| * 0.5$ // for $n = 1$;
 $R_{n1} = 0.75$
 $R_{n2} = R_{n-1} - |R_{n-1} - R_{n-2}| * 0.5$ // for $n = 1$;
 $R_{n2} = 0.25$
 if x_{th} for $R_{n1} > x_{th}$ for R_{n2} then
 $R_n = R_{n1}$, and $x_{th} = x_{th}$ for R_{n1}
 else
 $R_n = R_{n2}$, and $x_{th} = x_{th}$ for R_{n2}
 if $x_{th} \leq 1$ pixel then
 $R = R_n$, and go to OUT
 else
 go to Calculating R_n

OUT:
 $P_{xr} = P_{11-xr} + (P_{12-xr} - P_{11-xr}) * R$

ALGORITHM 3: Transformation grid (TG) algorithm.

of the size and position of the object bounded box that is generated by EKF prediction R_{kal} , or object appearance template T^i that was presented. The parameters used in the algorithm are briefly mentioned in Table 2.

As illustrated in Algorithm 4, the proposed algorithm not only defines the events and their response but also arranges them according to their priority in descending order. The priority of events means, the order in which the event is tested with respect to the other events. Accordingly, if one event for the object is triggered, other events will be discarded.

As illustrated in Algorithm 4, the algorithm works over each workstation using some inputs such as SSB which is produced using the current frame by workstation segmentation method. Another input to the algorithm is KBB, which is generated in the previous frame by extended Kalman filter. The number of elements in SSB and KBB sets may be different because of differing objects from the current frame to the previous one. $G_{kal} []$ is an input that comes from the server and contains the predicted location of all objects in the tracked environment. AT is a set of appearance templates of objects in the previous frame which are frequently updated in every frame. On the other hand, the tracking events

TABLE 2: Parameters used in the tracking algorithm.

Parameters	Description
R_{seg}^i	Rectangle bounded object i created by segmentation (background subtraction method and the other refining filters)
R_{kal}^i	Rectangle bounded object i predicted by EKF in previous frame
T^i	Appearance template of object i
G^k	The k th object at the workstation defined by its real position
$G_{array} []$	An array that contains all objects that were sent to the server by workstations
$G_{server} []$	An array that contains all objects created by server's association module.
$G_{kal} []$	An array that contains the predicted location of each object in $G_{server} []$, this prediction done by EKF in the server's module.

are arranged by their priority in a descending manner, as indicated in Figure 15. Moreover, the first three events (which are E_1 , E_2 , and E_3) depend essentially on R_{seg} . These

FIGURE 13: Workstations FOVs versus their illustration in the server coverage aid tool.

FIGURE 14: Three players are beginning tracking as one object, and then they split.

events are located in a loop to test the properties of each R_{seg} . In the same manner, the second three events depend essentially on R_{kal} . Responses of all events depend on two modules at each workstation and the server, which are WRM and SRM, respectively.

Triggering E_1 takes place whenever any element ϵ SSB presents a new object. This happens if any SSB element, R_{seg}^x , is not an object in the tracked environment by comparing it with all elements in G_{kal} []. The response of this event is R_1 , which guarantees the new object creation over both SRM and WRM.

The second event E_2 is tested for R_{seg}^x unless E_1 has triggered. Event E_2 monitors an object entrance into the workstation's FOV which may be triggered when two conditions are valid. The first condition is that the entering

object does not exist in the previous frame. The second is that the entering object must have existed previously in the tracked environment, as constrained the proposed tracked system previously, so it should be an element in G_{kal} []. The response of event E_2 is R_2 , which guarantees the entered object registration over the workstation.

The third event (E_3) detects an object splitting into two or more objects. It depends on the sudden varying of object size. Another condition is to ensure that the split object was not previously detected over the system by another workstation. Hence, split operation can be performed in a workstation by the server and by the other workstations feeds. The response (R_3) tries to detect the objects generated by split operation. The split objects may be known or unknown objects. The known objects mean that they were

FIGURE 15: The flow chart of tracking events algorithm.

(a) Grouping of two players in left frame is solved by another camera frame

(b) Frames from one camera solve two splits

(c) First row shows entering of a player, who takes the same label in another camera shown in the second row

(d) Frames show exiting of a player from a camera's FOV and system stopping of tracking him

FIGURE 16: Four cases show the robust of the tracking algorithm in solving some of tracking problems.

```

Inputs data:
SSB: Set of N objects Segmented Bounded Boxes
 $SSB = \{R_{seg}^1, R_{seg}^2, \dots, R_{seg}^N\}$ 
KBB: Set of M objects EKF Bounded Boxes
 $KBB = \{R_{kal}^1, R_{kal}^2, \dots, R_{kal}^M\}$ 
AT: Set of M Appearance Templates
 $AT = \{T^1, T^2, \dots, T^M\}$ 
Algorithm
Defining Events for each Workstation:
  For each  $R_{seg}^x \in SSB$ , test  $E_1$ ,  $E_2$ , and  $E_3$  in order
   $E_1$ :
    if  $R_{seg}^x$ .not_matched (one in  $G_{kal}[]$ )
    Then
      Trigger  $R_1$ 
      go to OUT1
   $E_2$ :
    if  $R_{seg}^x$ .not_matched (any  $R_{kal}^i \in KBB$ ) AND
        $R_{seg}^x$ .matched (one in  $G_{kal}[]$ )
    Then
      Trigger  $R_2$ 
      go to OUT1
   $E_3$ :
    if size ( $R_{seg}^x$ )  $\ll$  matched size ( $R_{kal}^i$ ) AND
        $R_{seg}^x$ .not_matched (one in  $G_{kal}[]$ )
    Then
      Trigger  $R_3$ 
      go to OUT1
  OUT1: Next
  For each  $R_{kal}^x \in KBB$ , test  $E_4$ ,  $E_5$ , and  $E_6$  in order
   $E_4$ :
    if  $R_{kal}^x$ .Outside (FOV) AND
        $R_{kal}^x$ .not_matched (any  $R_{seg}^i \in SBB$ )
    Then
      Trigger  $R_4$ 
      go to OUT2
   $E_5$ :
    if size ( $R_{kal}^x$ ) < matched size ( $R_{seg}^i$ ) AND
        $R_{kal}^x$ .intersected_with (any  $R_{seg}^y$ )
    Then
      Trigger  $R_5$ 
      go to OUT2
   $E_6$ :
    if  $R_{kal}^x$ .matched ( $R_{seg}^x$ )
    Then
      Trigger  $R_6$ 
      go to OUT2
  OUT2: Next
Events Responses:
Responses are done over:
SRM (Server Response Module) and WRM (Workstation
Response Module)
 $R_1$ :
  new WRM.register ( $G^k$ )
  WRM.create_object_template ( $G^k$ )

```

ALGORITHM 4: Continued.

```

WRM.send_to_SRM ( $G^k$ )
SRM.associate ( $G_{array}[]$ ) get  $G_{server}[]$ 
SRM.create_and_send_to_each_WRM ( $G_{kal}[]$ )

R2:
new WRM.register ( $G^k$ )
WRM.send_to_SRM ( $G^k$ )
SRM.associate ( $G_{array}[]$ ) get  $G_{server}[]$ 
SRM.create_and_send_to_each_WRM ( $G_{kal}[]$ )

R3:
For all  $R_{seg}^y$ 
  if not  $R_{seg}^y$ .matched_any_of ( $R_{kal}^i$ ) AND  $R_{seg}^y$ .near ( $R_{seg}^x$ )
  Then
     $G^v$  = WRM.get_object_of ( $R_{seg}^x$ )
    if  $G^v$ .number_of_contained_objects == 1
    Then
      WRM.trigger_new_response ( $R_{seg}^y$ )
      WRM.build_history_of ( $R_{seg}^y$ )
    Else if  $G^v$ .number_of_contained_objects > 1
    Then
       $R_{seg}^y$ .associate_to_objects_in ( $G^v$ )
  Next
WRM.trigger_stable_response ( $R_{seg}^x$ )

R4:
WRM.stop_tracking ( $G^k$ )

R5:
WRM.register_group ( $G^k$ ) =  $G^v$  +  $G^w$ 
WRM.announce_to_SRM ( $G^k$ )
SRM.associate ( $G_{array}[]$ ) get  $G_{server}[]$ 
SRM.create_and_send_to_each_WRM ( $G_{kal}[]$ )
if WRM.search( $G_{server}[], G^v, G^w$ ) == True
Then
  WRM.split ( $G^k$ )
Else
  WRM.give_EKF_position ( $G^v, G^w$ )

R6:
WRM.track ( $G^k$ ); WRM.update_EKF ( $G^k$ ); WRM.update_template ( $T^x$ )
WRM.send_to_SRM ( $G^k$ )
SRM.associate ( $G_{array}[]$ ) get  $G_{server}[]$ 
SRM.create_and_send_to_each_WRM ( $G_{kal}[]$ )

```

ALGORITHM 4: The tracking events algorithm.

merged after starting tracking, so they were known. On the other hand, the unknown objects are merged before starting tracking. Although the algorithm can solve the two cases separately in an efficient way, it may fail when the object splits into known and unknown objects, object splits in complicated split-merge situations, or object splits in a crowded area. Finally, the newly unknown objects will be registered at the tracking server and at all workstations by triggering R1.

The second set of events (which are E_4 , E_5 , and E_6) depends on KBB set. E_4 is triggered when an object exits the workstation's FOV. Two conditions ensure the objects exiting from the FOV, which guarantees the correctness of EKF prediction. The response of event R_4 is to stop tracking such exited object. To stop tracking an object means deleting its KBB element and not to build G^k for it.

The merge situation of objects fires the event E_5 . Triggering E_5 depends on a sudden change of an object size to larger size and the prediction of intersecting elements in KBB. The response R_5 is to register the produced merged object as a group of its constituting objects. SRM tries to split the merged objects using the knowledge that comes from other workstations.

All the previous events are addressed as nonstable events, so the objects that do not trigger these events are called to be tracked in a stable way. The last event, E_6 , expresses this situation when an element from KBB matches one of SBB as in traditional way of tracking. The response of event R_6 is to match the object track, update EKF, and update AT element of this object.

Once all the possible events are successfully triggered, the continuity of objects tracks can be done correctly. Figure 16

FIGURE 17: The tested five paths and two projection planes in the middle camera.

TABLE 3: Experiment I results; comparing five paths using the image transformation algorithm and two projection planes. The comparison employs the RMSE over the two axes.

	RMSE	Path 1	Path 2	Path 3	Path 4	Path 5
TG	X-axis	0.59 m	0.51 m	0.60 m	0.31 m	1.02 m
	Y-axis	0.65 m	0.57 m	0.67 m	0.28 m	0.98 m
Plane 1	X-axis	29.77 m	13.53 m	28.81 m	19.13 m	31.35 m
	Y-axis	33.03 m	15.28 m	25.11 m	18.64 m	35.21 m
Plane 2	X-axis	12.95 m	14.69 m	11.73 m	10.75 m	15.02 m
	Y-axis	13.48 m	15.49 m	13.22 m	9.61 m	13.84 m

explores some situations that were faced by the system with the corresponding triggered events, solved problems, and server duties.

5. Experiments and Results

In this section, three experiments will be presented. The first two experiments determine the accuracy of the proposed image-to-world transformation method, while the third evaluates the proposed algorithm of object tracking.

5.1. Experiment I: Measuring the Accuracy of Image-to-World Transformation Algorithm. In this experiment, the efficiency of the proposed image-to-world transformation method will be measured on a real three-dimensional CAD model. This model simulates the soccer pitch with a camera's FOV that captures the center of the pitch. The frame size was adjusted to be 720×576 pixels. Each point in the TG points is 8 meters far from its neighbors in the four directions. Five paths are constructed in the CAD simulation, each of them consists of 40 points. Each point real position is known in the CAD simulation, and its image position is manually fed. We choose such paths, as well as their points positions, to test the efficiency of the algorithm near and far, boundaries, and middle of the camera's FOV. Moreover, we did not take any diagonal (over the camera's FOV) path that may get tricky results with the calculated Root Mean Square Error (RMSE). The proposed transformation method is compared here with the fast projection method [17] using two projection planes. Each plane is constructed by its four corners which present four noncollinear points P_{11} , P_{12} , P_{22} , and P_{21} .

TABLE 4: Experiment III results; the event-tracking evaluation.

observing	Manual and CAD	System succession	Event_ratio
New object register event E_1	25	22	88%
Object-enter event E_2	31	30	97%
Object-split event E_3	34	27	79%
Object-exit event E_4	26	26	100%
Object-merged event E_5	32	30	94%
Targets trajectory correlation	1.0	0.81	81%

Transformation using each plane depends on (4) which was discussed previously. Figure 17 shows the CAD simulation with the constructed paths and planes.

For each path, RMSE is calculated using each point's real position (P_{real}) and image position (P_{image}) for N points using the following formulas for the two real axes X and Y :

$$\text{RMSE}_X = \sqrt{\frac{1}{N} \sum_{i=1}^N (P_{\text{real}-X} - P_{\text{image}-x})^2}, \quad (5)$$

$$\text{RMSE}_Y = \sqrt{\frac{1}{N} \sum_{i=1}^N (P_{\text{real}-Y} - P_{\text{image}-y})^2}.$$

Results for each path are summarized in Table 3. The transformation algorithm calculations for one point took less than 1 millisecond on an Intel-I5 CPU, and the calculations module was built over. NET framework and ran under Windows 7 operating system.

Results in Table 3 show the robustness of the proposed image-to-world transformation algorithm. The RMSE is between 0.28 m and 1.02 m in a TG of 8 m points distance. These results lead to an error between 3.5% (in the higher resolution areas) and 12.7% at most (in the poor-resolution areas). Moreover, any transformed point will retain in the equivalent real cell of TG after transformation. In other words, the transformed point will not jump to another cell in real grid after transformation, and so the algorithm will not get any tricky results.

5.2. Experiment II: Extending the Image-to-World Transformation Algorithm. This experiment extends the previous one by assuming that each TG point is 8, 16, 24, 32, and 40 meters far from its neighbors in the four directions. The RMSE of the same five paths shown in Figure 17 is calculated, and the results are illustrated in Figure 18.

By observing Figure 18, it is concluded that the proposed transformation algorithm outperforms fast projection method, referring to results in Table 3. Moreover, the RMSE remains in an acceptable range even at large distances between the TG points.

5.3. Experiment III: Evaluating the Event-Tracking Algorithm. In this experiment, the proposed tracking algorithm will

FIGURE 18: RMSE of the five paths tested in experiment II. The test was performed over five different distances of TG points.

FIGURE 19: Workstations' FOVs versus the illustration of them in server coverage aid tool.

be tested over a three-dimensional CAD simulation. The environment of this experiment is the soccer pitch covered by a network of five cameras. The pitch includes 25 moving objects, which are divided into two teams and three referees (with different color clothes). The simulation period was 2 minutes length, 25 frames per second, MPEG format, and 768×576 pixels in frame size. The workstations, as well as the server, are Intel-I5 CPU, and the modules were built over .NET framework running under Windows 7 operating system. The FOVs of the five static cameras used to cover the soccer field are shown in Figure 19.

The proposed tracking events algorithm can be evaluated by comparing the events generated automatically by the system versus the manual observation of workstations and server events as percentage form. The event ratio can be calculated using the following formula:

$$\text{Event}_{\text{ratio}}(\%) = \frac{\text{Events}_{\text{generated by system}}}{\text{Events}_{\text{manually observed}}} \% \quad (6)$$

The evaluation results are detailed in Table 4.

The reported results in Table 4 show the success of event-tracking algorithm in most cases. The results verify that the algorithm was able to handle the continuity of objects tracks well, which is the main goal of any tracking system.

As a proof of system ability to preserve the continuity of objects tracks, we present two plots of objects tracks with

two different ways. The first one is a plot of two objects over the pitch as shown in Figure 20. The detected events by the event-tracking algorithm are included along each object track. Moreover, the plotting compares the real object track (path) with the one generated by the system, which insures the good performance and robustness of the system. It is also noted that the objects tracks (paths) are smoothed by Kalman filter.

6. Conclusions

In this paper, a novel strategy was presented for constructing a tracking system in a crowded environment. The novelty of this strategy appeared in the flexibility of the system architecture, clearly solving image transformation and system setup problems, exploring a new model of the tracked objects, and finally presenting an event-driven algorithm for tracking objects over a network of static cameras. The proposed tracking strategy provides a potential to the surveillance system which requires wide area observation and tracking objects over multiple cameras, for example, airport, train station, or shopping malls. It is not possible for a single camera to observe the complete area of interest as the structures in the scene constraint the visible areas and the devices resolution is restricted. Therefore, the surveillance system of wide areas requires multiple cameras for objects tracking. Over a CAD of soccer game, experiments have

FIGURE 20: Comparing real and system path of two players. The events new (N), merging (G), and splitting (S) are shown on the path.

shown that the proposed tracking strategy performs well even in cluttered environments. Our future investigations are to extend the proposed tracking algorithm to include other events. Moreover, although the soccer game was a very rich tracking environment, the system should be tested in other environments such as parking areas, other sports playgrounds, and hyper markets.

References

- [1] C.-C. Chia, W.-K. Chan, and S.-Y. Chien, "Cooperative surveillance system with fixed camera object localization and mobile robot target tracking," in *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology (PSIVT '09)*, vol. 5414 of *Lecture Notes in Computer Science*, pp. 886–897, Springer, Tokyo, Japan, 2009.
- [2] W. Niu, L. Jiao, D. Han, and Y.-F. Wang, "Real-time multi-person tracking in video surveillance," in *Proceedings of the 4th IEEE Pacific Rim Multimedia Conference (PCM '03)*, pp. 1–5, Singapore, December 2003.
- [3] M. Tändl, T. Stark, N. Erol, F. Löer, and A. Kecskeméthy, "An object-oriented approach to simulating human gait motion based on motion tracking," *International Journal of Applied Mathematics and Computer Science*, vol. 19, no. 3, pp. 469–483, 2009.
- [4] D.-X. Lai, Y.-H. Chang, and Z.-H. Zhong, "Active contour tracking of moving objects using edge flows and ant colony optimization in video sequences," in *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology (PSIVT '09)*, vol. 5414 of *Lecture Notes in Computer Science*, pp. 1104–1116, Springer, Tokyo, Japan, 2009.
- [5] B. Sugandi and H. Kim, "A moving object tracking based on color information employing a particle filter algorithm," in *Proceedings of the 14th International Symposium on Artificial Life and Robotics*, Oita, Japan, 2009.
- [6] J. Kang, I. Cohen, and G. Medioni, "Soccer player tracking across uncalibrated camera streams," in *Proceedings of the IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (PETS '03)*, Nice, France, October 2003.
- [7] M. Mozerov, A. Amato, X. Roca, and J. González, "Solving the multi object occlusion problem in a multiple camera tracking system," *Pattern Recognition and Image Analysis*, vol. 19, no. 1, pp. 165–171, 2009.
- [8] O. Javed and M. Shah, "Tracking and object classification for automated surveillance," in *Proceedings of the 7th European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.
- [9] L. J. Latecki and R. Miezianko, "Object tracking with dynamic template update and occlusion detection," in *Proceedings of the 18th IEEE International Conference on Pattern Recognition (ICPR '06)*, vol. 1, pp. 556–560, Hong Kong, August 2006.
- [10] T. Ellis, D. Makris, and J. Black, "Learning a multi-camera topology," in *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 165–171, Nice, France, October, 2003.
- [11] D. Setterwall, *Computerized video analysis of football—technical and commercial possibilities for football coaching*, M.S. thesis, Department of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden, 2004.
- [12] M. Xu, J. Orwell, L. Lowey, and D. Thirds, "Architecture and algorithms for tracking football players with multiple cameras," *IEEE Proceedings: Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 232–241, 2005.
- [13] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.
- [14] C. J. Needham and R. D. Boyle, "Performance evaluation metrics and statistics for positional tracker evaluation," in *Proceedings of the International Conference on Vision Systems (ICVS '03)*, pp. 278–289, Graz, Austria, 2003.
- [15] M. Xu, J. Orwell, and G. Jones, "Tracking football players with multiple cameras," in *Proceedings of the International Conference on Image Processing (ICIP '04)*, vol. 5, pp. 2909–2912, Singapore, October 2004.

- [16] R. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '86)*, pp. 364–374, Miami Beach, Fla, USA, 1986.
- [17] S. Iwase and H. Saito, "Tracking soccer players based on homography among multiple views," in *Visual Communications and Image Processing*, vol. 5150 of *Proceedings of SPIE*, pp. 283–292, Lugano, Switzerland, July 2003.
- [18] S. Iwase and H. Saito, "Tracking soccer player using multiple views," in *Proceedings of the 8th IAPR Workshop on Machine Vision Applications (MVA '02)*, Nara, Japan, December, 2002.
- [19] J. B. Hayet, J. H. Piater, and J. G. Verly, "Fast 2D model-to-image registration using vanishing points for sports video analysis," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '05)*, pp. 417–420, Genoa, Italy, September 2005.
- [20] J. Sullivan and S. Carlsson, "Tracking and labelling of interacting multiple targets," in *Proceedings of the 9th European Conference on Computer Vision (ECCV '06)*, vol. 3953 of *Lecture Notes in Computer Science*, pp. 619–632, Graz, Austria, May 2006.
- [21] S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Visual Communications and Image Processing*, vol. 5308 of *Proceedings of SPIE*, pp. 881–892, San Jose, Calif, USA, January 2004.
- [22] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.
- [23] P. Remagnino, A. Baumberg, T. Grove et al., "An integrated traffic and pedestrian model-based vision system," in *Proceedings of the 8th British Machine Vision Conference (BMVC '97)*, A. F. Clark, Ed., pp. 380–389, Essex, UK, 1997.
- [24] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.
- [25] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 918–923, 2003.
- [26] R. Rosales and S. Sclaroff, "Improved tracking of multiple humans with trajectory prediction and occlusion modeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '98)*, Santa Barbara, Calif, USA, June 1998.
- [27] N. D. Doulamis, *Coupled Multi-Object Tracking and Labeling for Vehicle Trajectory Estimation and Matching*, Springer Science+Business Media LLC, 2009.
- [28] A. Farina and S. Pardini, "Track-while-scan algorithm in a clutter environment," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 14, no. 5, pp. 769–779, 1978.
- [29] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, 1988.
- [30] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.
- [31] S. Blackman, *Multiple Target Tracking with Radar Applications*, Artech House, 1986.
- [32] L. Klein, *Sensor and Data Fusion Concepts and Applications*, Bellingham, Wash, USA, SPIE Optical Engineering Press, 1993.
- [33] A. Gad and M. Farooq, "Tracking highly maneuvering targets in clutter using interacting multiple model fuzzy logic based tracker," in *Signal Processing, Sensor Fusion, and Target Recognition XI*, vol. 4729 of *Proceedings of SPIE*, pp. 23–34, Orlando, Fla, USA, April 2002.
- [34] A. Gad and M. Farooq, "Viterbi-based data association techniques for target tracking," in *Signal Processing, Sensor Fusion, and Target Recognition XII*, *Proceedings of SPIE*, pp. 37–46, Orlando, Fla, USA, April 2003.
- [35] F. White, "A model for data fusion," in *Proceedings of the 1st International Symposium on Sensor Fusion*, 1988.
- [36] M. Bedworth, "Probability moderation for multilevel information processing," DRA Technical Report DRA CIS(SEI)/651/8/M94.AS03BPO32/1, 1994.
- [37] M. Bedworth and J. O'Brien, "Omnibus model: a new model of data fusion?" *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 4, pp. 30–36, 2000.