Crowd Scene Tracking

Michael Feist
mdfeist@ualberta.ca

Tamara Bain
tbain@ualberta.ca

Maciej Ogrocki
ogrocki@ualberta.ca

Benjamin Lavin
blavin@ualberta.ca

I. INTRODUCTION

Automated detection and tracking pedestrians in crowds is a highly studied facet of computer vision, both due to its complexity and its many different applications. The large amount of research has resulted in a diverse range of successful approaches, so that the question of detection and tracking can be answered successfully in more than one way. Approaches can be broken down generally into two sub genres, macroscopic crowd capture solutions, and microscopic crowd capture solutions [19][20]. Macro crowd capture involves looking at the crowd as a whole and utilizes holistic properties of the scene, such as the overall density and flow of moving objects, to determine motion vectors of the crowd [20]. This approach is popular when simulating crowd scenes for videos or movies, or analyzing the response of crowds to events such as natural disasters. In contrast, microscopic crowd detection and tracking techniques focus on the individual pedestrians within the crowd. Because of the intricacies associated with tracking individuals through a visually cluttered space this method is often more time consuming to implement, but results in more accurate information about the scene and pedestrians at hand.

Crowd tracking stems from the broader concept of objecting tracking and the difficulties faced are similar to those seen in any object tracking problem. Difficulties can arise due to abrupt object motion, changing appearance patterns of both the object and the scene, nonrigid object structures, object-to-object and object-to-scene occlusions, and camera motion [19]. These difficulties are more often seen in microscopic crowd tracking techniques because the number of objects being tracked is higher, and the tracked objects are much smaller relative to the scene.

Both macro and micro crowd tracking solutions have been used in previous literature with varied results. Both techniques have advantages and pitfalls.

II. MACRO CROWD CAPTURE TECHNIQUES

A. Overview of Macro Crowd Capture and Simulation

Macro crowd capture focuses on gathering information from the crowd as a whole, viewing the crowd as a single fluid entity and relying on data such as: flow, density, scene geography, social forces, and event forces [1][2][5][12]. Rather than apply motion capture data to each individual, each individual is classified into a group and is applied a general motion capture file (such as strolling or chatting) [3]. Macro crowd motion capture has been used to create a realistic crowd that behaves similarly to a real crowd without precise recording of each individual in the crowd. This avoids many of the problems of detection while remaining accurate enough for a wide variety of applications. It has been used to create realistic crowds in movies, video games and other simulations [1][5]. By detecting specific changes in flow and speed of a crowd it has also been used to computationally detect crowd related emergencies automatically [2]. This can be expanded to detect abnormal behaviours in crowds [12]. Macro crowd motion capture is often a combination high and low-level behavior analysis [3]. High level behavior analysis looks at information pertaining to the entire crowd such as the flow of the crowd and average speed of the crowd. This type of behaviour has been used to find where a crowd is moving, locations of higher and lower density, and points of entry and exit [1][2][5][6].

In contrast, low-level behaviour analysis focuses on groups or individuals within a crowd. This has been used to take data on individual motion, classify it, and apply preset motion capture data based on the classification [3][12].

B. High Level Behavior Analysis

The starting point for high level analysis is flow and speed of the crowd [1][2][5]. N. Courty et al., B Boghossian et al., and R.Mehran et al. all used variations on the optic flow algorithm to detect directions of movement in a crowd and use this as a starting point for further analysis and simulation generation. R. Mehran et al. supplemented the optic flow algorithm by applying the social force model to improve their crowd simulation [5]. Social force can be described as the behaviour of the crowd as a result of the actions of individuals [5]. By computing social force, R. Mehran et al. were able to determine the ongoing behavior of the crowd and its movement towards a goal or destination [5]. One high level approach for tracking in high density crowds is to use a scene structure based force model[17]. S. Ali and M. Shah used describe their scene structure force model by proposing that individuals moving about a scene are subject to local and global forces that influence the individuals locomotive behaviour. Three floor fields are described which determine the individuals probability of movement from one location to another [17]. The Static Floor Field (SFF) is the attraction field. This represents the typical crowd motion toward attractive locations in the environment, for example, the exit door. The Boundary Floor Field (BFF) represents the repulsive forces that shape the motion paths of crowds such as fences, walls, or other obstacles. Finally, the Dynamic Floor Field (DFF) is representative of the immediate behaviour of other sections

of the crowd in the vicinity of the specific section being analysed. The collective behavioural patterns represented by these fields can be combined then constrain the likely locations or paths that can be taken by the target in the scene [17]. An analysis of high level behavior has been shown to be sufficient for detecting abnormalities in crowds [5][9]. However such analysis falls short of pointing out what the abnormal behavior or actions actually look like, such detail requires a low-level analysis of the crowd behavior.

C. Low Level Behavior Analysis

Several types of low level behavior analysis have been used to increase detail in crowd simulations resulting from macro motion capture [3]. K. Lee et al. looked at individuals in a crowd and classifies them by the type of action they were performing. By applying preset motion capture data based on these classifications, the detail in the resultant crowd simulation was improved [3]. Using this system it was possible to have simulated individuals perform specific actions such as dancing and cheering [3]. As opposed to the high level behaviour studies in which the individuals only simply walk [1][5]. A major issue with low level behaviour is that it uses pre recorded motions that are then applied to a models using the information gathered from the input video. By doing this multiple individuals will perform the same actions causing a clone effect [4]. The most optimal way to fix this is to have many pre recorded motions; this however, is expensive. R. McDonnell et al. use proximity and model typing to create heterogeneous crowds without having to increase the amount of animations [4]. The algorithm described in S. Saxena et al. used KLT (Kanade-Lucas-Tomasi) to track feature points. Using KLT is handy because it automatically gives the 2-dimensional motion vector for each feature point. After tracking the feature points, the next step is to clean up the data. To clean up the data, first feature points that remain stationary are removed as these points are most likely due to noise or belong to background objects. Next, points that are relatively similar in location and direction are clustered together. The method in S. Saxena et al. tracks the crowds principal direction, speed, and mobility. Direction is calculated by using a histogram of all the motion vectors for a cluster. A histogram is used in order to determine if a crowd is showing abnormal behavior; this occurs when the cluster has multiple directions. Speed is measured by analysing the change in the location of the crowd. For this approach to work, the camera needs to be calibrated. The calibration involves the use of two pairs of parallel lines (perpendicular to each other) on the ground and the height of a reference object [12]. The authors do not go into detail of how mobility is calculated, but they seem to simply track the change in the number of feature points for each cluster. This attribute can thus be used to determine abnormal events if the number of valid feature points changes drastically [12]. A.B. Chan et al. proposed a method that is able to track individuals without using object/pedestrian detection or feature tracking. First the scene is segmented into crowds with different motions [14]. Next features are extracted from each crowd segment.

Finally, the number of people per segment is estimated with Gaussian process regression.

D. Limitations

High level behaviour can be used to detect the flow and behaviour of a crowd but will lack the specific details that can be found in tracking and capturing the actions and movement of each individual. R. Mehran et al. demonstrated simulated crowds moving in a realistic approximation of a crowd but are unable to directly capture information of a real world crowd and simulate it [5]. In fields such as risk assessment or emergency procedure evaluation, where individual actions are valuable, a more detailed analysis than can be afforded by macro crowd capture should be used. One downfall to the method proposed in cacro crowd capture is that it groups people together as opposed to tracking each individual person. This approach makes it difficult to determine crowd density. R. Mehran et al. do not seem to address this issue, but they do explain previous work in the field that has solutions to crowd density [1][5][12]. The algorithm proposed by A.B. Chan et al. has the ability to give us an accurate density of the crowd. However, the method requires that training be run on a subset of the data. In the paper, the authors described how they first trained on 800 frames and then ran the testing on 1200 frames [14]. Because of the need for training, one could not simply grab utilize a feed from any camera source and expect the algorithm to work instantaneously. Instead, the user must manually train the method to estimate the density, a very time consuming process.

III. MICRO CROWD CAPTURE TECHNIQUES

A. Overview of Micro Crowd Capture

In contrast to the macro techniques described above, micro crowd motion capture eschews simulation of a crowd for detection and tracking of distinct individuals. Many studies follow a similar basic algorithm in this regard, starting with object detection of pedestrians, and following with some method of object tracking [6][7][8][9][10]. The major challenges of micro crowd capture are occlusion, adequate detection, false positives, and perspective. Most studies attempt to supplement their detection and tracking with some additional algorithm or technique to address challenges such as occlusion, false positives, and varying camera perspective. Despite the highly successful nature of many of these studies, they are still limited by large training sets, and may be situational applicable depending on the density of the crowd or the number of cameras used.

B. Object Detection

Pedestrian-detection is often the first step in micro crowd capture techniques. In dense crowds two hurdles for accurate detection are large areas of occlusion and varying perspectives of moving objects. Many studies address occlusion of pedestrian bodies by focussing on head detection [6][7][8][10]. The Viola and Jones Haar-like AdaBoost cascade detector is cited frequently as the preferred method of object detection and

is trained to classify pedestrian heads [7][8][10]. Using this method requires large training sets of images (4000+) as well as half as many negative images. The large size of the training set is partially due to the multiple potential perspectives of the human head as a pedestrian moves throughout the scene. Each potential perspective and distortion of a human head must be trained for to ensure high detection rates. As an alternative to the Viola and Jones detector and to address these potential variations in viewpoint, Rodriguez et al. utilizes a parts based model which has improved detection accuracy on deformed datasets [11]. Object detection is a balance between adequate detection and the reduction of false positives. After settling on an adequate detection method many studies apply a secondary measures to reduce false positives such as scene geometry, energy expenditure, or crowd density [6][7][8][10]. Scene geometry is an easy way to improve detection results. By noting the average position of heads within a scene, and knowing the average size of a human head, the position of the Head Plane can be calculated. Using a head plane and calculating the expected size for each head within that plane makes it possible to take depth into account in single camera scene captures [6][10]. Other studies use scene geometry to surmise most-likely paths of motion in a scene [6][7]. D.Zhang et al. favors detection of objects occurring along likely scene trajectories [7]. These enhancements to object detection can be applied in succession to refine results. M. Rodriguez et al. uses both scene geometry and crowd density to reduce false positives and improve head detection in crowds. Their detection rate and number of false positives are better when both methods are used [6].

C. Single Camera Tracking

Occlusion is one of the major problems when tracking individuals with single cameras in high density scenes. The method proposed by V.K. Singh et al. tries to solve this issue [15]. The tracking problem is divided into two stages. In the first stage, a tracking algorithm tracks individuals and, if it briefly loses track of an individual, simply tracks the reappearance as a new individual. In the second stage, the algorithm attempts to combine multiple motions together which belong to the same individual. D. Zhang et al. manage occlusion during tracking by devising a set of heuristics for favoring energy conservation within a scene. These heuristics filter out trajectories that would place two pedestrians on the same path at the same time, have pedestrians appear suddenly in the middle of the scene or have pedestrians greatly vary their speed/position within a short period of time [7]. This method improves the robustness of tracked trajectories over multiple iterations.

D. Multi Camera Tracking

An idea proposed to help track dense crowds and handle occlusion is to use multiple cameras that overlap the scene instead of a single camera [13][16]. R. Eshel and Y. Moses propose one of these methods [13]. The issue with their method is that their algorithm assumes a set of synchronized

Fig. 1. Results of the density algorithm. Green for areas with low density, yellow for areas of medium density, and red for areas of high density.

and partially calibrated cameras overlooking a single scene, where head tops are visible. With their setup they are able to detect a persons head without using any object detection. This is achieved by applying a homography to map all the cameras to a unique plane that is parallel to the ground and at the height of the persons head. This means the same point on the plane will map to the same image location in all cameras. If an object does not fall on the plane then it wont align in the multiple views after the homography is applied. Because of this they are able to isolate the head by looking at what moving objects align on the plane. Once the head is tracked, the speed and direction can be estimated using the previous frame [13]. If the head being tracked does not move relatively in the same motion, then it is treated as a false positive and discarded. Finally, it is important to note that the results improve as more cameras are added [13]. Better results are seen once 5 cameras have been added, however once the number of cameras reaches 7, the benefits of adding further cameras are negligible. M. Liem and D. Gavrilas method is similar to R. Eshel and Y. Mosess algorithm. Both algorithms try to recreate a 3D environment [13][16]. However, M. Liem and D. Gavrilas method first uses volume segmentation to try and separate a person, followed by a combination of feature points and likelihood functions to predict which volumes are individual people [16].

IV. COMBINING MICRO AND MACRO CROWD CAPTURE

Several studies combine macro and crowd capture to refine their results. Rodriguez et. al use macro analysis of crowd density to improve head detection and tracking of individual pedestrians [6]. In contrast F. Zhao et. al use micro analysis of a pedestrian scene to supplement a macro style detection of abnormal activity [7]. Studies like these show that micro and macro crowd capture are not mutually exclusive and can be used in tandem to improve the accuracy of results.

V. IMPLEMENTATION

A. Density

One crowd parameter that was useful in other areas of the project was crowd density. In order to calculate the density of

a crowd we need to first make some assumptions. The first is that the people in the crowd are all moving roughly the same speed. We also assume that the size of a person stays roughly the same as they move through the scene. Finally we assume that the more an area in the scene is changing the more people are in that area.

With these assumptions we are able to calculate the density of a crowd by calculating the temporal difference. To calculate the temporal difference we simply take the current frame and subtract it from the previous frame. We then take the difference and create a threshold image. We found that a change in magnitude greater than 20 was significant enough to warrant valid change. Usually you dont want to include all changes in your threshold image since then the threshold would include noise from the camera. Next we divide the scene into smaller blocks. For scenes like the ones shown in Fig. 1 we found block sizes of 8 by 8 worked well. Next we iterate over the blocks and calculate the percentage of change. We then add the percent of change times some growth rate to the density map. Also it is important to decrease the density of the blocks over time.

$I_t \leftarrow |Im[x, y, t+1] + Im[x, y, t]|$
$I_{threshold} \leftarrow I_t > threshold$
$blocks_x = Im_{width}/block_{size}$
$blocks_y = Im_{height}/block_{size}$
for $i := 0; i < blocks_x; i{+}{+}$ **do**
    for $j := 0; j < blocks_y; j{+}{+}$ **do**
        $density[i, j] \leftarrow I_{density}[i, j] - decay\ rate$
        if $density[i, j] < 0$ **then**
            $density[i, j] \leftarrow 0$
        end
        $x1 \leftarrow i(block_{size})$
        $x2 \leftarrow i(block_{size}) + block_{size}$
        $y1 \leftarrow j(block_{size})$
        $y2 \leftarrow j(block_{size}) + block_{size}$
        $block \leftarrow I_{threshold}[x1 : x2, y1 : y2]$
        $density[i, j] \leftarrow$
        $density[i, j] + \alpha \sum_x^N \sum_y^N block[x, y]$
        if $density[i, j] > 1$ **then**
            $density[i, j] \leftarrow 1$
        end
    end
end

Algorithm 1: Density Calculation

B. Optic Flow

Optic Flow was used in other papers to calculate the flow of the crowd [1][2][5]. Most of these papers used the calculated flow information in a simulator. However, since we did not have a simulator the flow information of the crowd was not as useful to us. That being said we still were easily able to calculate the flow of the crowd using OpenCVs built-in Optic Flow function.

Fig. 2. Results of the Optic Flow algorithm. Small lines point in the direction in which the pixels are found to be moving. In this image you can see that the hand is moving very slightly from left to right.

```
flow =
    cv2.calcOpticalFlowFarneback(prev_gray,
    gray, None, 0.5, 3, 20, 3, 5, 1.2, 0)
```

There are a few issues with using Optic Flow to calculate the flow of the crowd. The first is that when two object are moving against each other they will cancel out the flow of both objects. Secondarily flow is not calculated equally over all objects in the crowd. Larger objects will have a greater in pack on the flow of the crowd as they will take up more space and appear as multiple pedestrians. As such videos that contain large non-pedestrian objects such as cars will be less accurate.

C. Affine and Metric Rectification

Following the solution proposed in section 2.7 of Hartley and Zisserman book, Multiple View Geometry, we were able to implement a two pass Metric Rectification algorithm. The first step is the user selects two pairs of parallel lines. Using theses lines we are able to apply an affine rectification using the following equations.

$$v_1 = m \times m' \qquad (1)$$

$$v_2 = n \times n' \qquad (2)$$

$$l = v_1 \times v_2 \qquad (3)$$

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \qquad (4)$$

Where m and m' are one set of parallel lines and n and n' are the other set of parallel lines.

Next we have the user select two pairs of orthogonal lines in the affine rectified image. Using the orthogonal lines we are able to retrieve a metric rectified image from the affine. Finally we allow the user to scale, rotate, and translate the results to allow them to get a better viewing experience. When the user

is happy with the results they can output the matrix to the screen allowing them to use the homography matrix in the tracking algorithms described below.

D. Tracking With Lucas-Kanade

One technique used to track people in a crowd was an implementation of Lucas-Kanade that tracked feature points. We begin by searching through the first frame in the video feed to find feature points. These feature points are areas in the image where corners are found. More specifically we used Shi-Tomasi corner detection. After we have our points we create a new track for each point.

We then start the main loop of the tracking algorithm. The loop starts by grabbing the next frame in the video feed and then trying to find the new locations of the feature points by using Optic Flow.

```
# The variable p0 is the original location of
    the points and p1 is the new location
p1, st, err =
    cv2.calcOpticalFlowPyrLK(prevgray, gray,
    p0, None, **lk_params)
```

Since people can move in and out of the frame we added a section of code that would add new feature points to the scene. This works by first calculating areas of change between the current frame and the previous frame. Next we search for new feature points in these areas of change. Finally we add the new feature point to our tracking points if that new feature point does not have a similar corresponding tracking point. These are points in relatively the same location.

We also try to remove points from the list of feature points, p1, that are no longer valid. The points that are removed are points that no longer move or move an unrealistic amount. The main issue with removing points that remain still for an extended period of time is that we can easily remove points that belong to a person that has briefly stopped moving and will continue moving. Because of this we need to assume that people are not stopping for this algorithm to work correctly.

After we have our updated list of feature points, p1, we need to add the points to their corresponding track. This posed some initial difficulties since the size of p1 is constantly changing and the indexed points dont always correspond to the same indexed track. What we mean by this is that the point in p1 at index i does not always belong to the track at index i. To keep track of all the changing information we created four arrays.

1) p1: An array of all the active points. These are the points currently being tracked by Lucas-Kanade.
2) st: An array of which points in p1 are no longer active and were removed from p1 during this last iteration. It is a bit array where 1 means that the point is still active and 0 means that the point was deleted.
3) tracks: A 2D array storing the path that each point tracks. It is an m x n array where m is the number of frames and n is the number of points.

Fig. 3. Results of our trained Haar Cascade that is detecting people. Video captured from the University of Alberta LRT station.

4) tracks_index: Since p1 and tracks dont match in size or indexing, we need tracks_index to keep a map between p1 and tracks.

The tracks_index array is updated by first initializing an array with values 0 to n-1, where n is the number of points in p1. Each iteration we obtain a list of the points which were deleted from p1, st. We then iterate over st and, if a point is removed, we set the value in tracks_index corresponding to the index in st to -1, and subtract one from all the remaining values in tracks_index.

After this step, if p1 has more points than the max value in tracks_index, it means that an additional point was added. In this case we append the max value plus one to the end of tracks_index.

Finally, we go through p1 and find the corresponding index in tracks_index. We then add the point in p1 to the correct path in tracks, given the found index in tracks_index.

After we have finished recording all the tracks we do a couple passes to clean up the data before we output it to a file. For example we delete tracks that we deem to short and were probably just some random blip. We also remove tracks that dont seem to follow a consistent path.

E. Object Detection

To accurately track a person we need to be able to identify what objects in our scene are people.

Fig. 4. Results of the tracking algorithm that utilizes the Object Detection.

Fig. 6. Results of the Unity crowd playback data outputted from the tracking algorithms.

H. Unity

VI. RESULTS

VII. DISCUSSION

REFERENCES

[1] N. Courty and T. Corpetti, 'Crowd motion capture', *Computer Animation and Virtual Worlds,* vol. 18, no. 4-5, pp. 361-370, 2007.

[2] B. Boghossian and S. Velastin, 'Image Processing System for Pedestrian Monitoring Using Neural Classification of Normal Motion Patterns', *Measurement and Control,* vol. 32, no. 9, pp. 261-264, 1999.

[3] K. Lee, M. Choi, Q. Hong and J. Lee, 'Group Behavior from Video: A Data-Driven Approach to Crowd Simulation', *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2007),* pp. 109-118, 2007.

[4] R. McDonnell, M. Larkin, S. Dobbyn, S. Collins and C. O'Sullivan, 'Clone attack! Perception of crowd variety', *ACM Trans. Graph.,* vol. 27, no. 3, p. 1, 2008.

[5] R. Mehran, A. Oyama and M. Shah, 'Abnormal Crowd Behavior Detection using Social Force Model', *Computer Vision and Pattern Recognition,* vol. 2009, pp. 935-942, 2009.

[6] M. Rodriguez, I. Laptev, J. Sivic, and J. Audibert 'Density-aware person detection and tracking in crowds', *Computer Vision (ICCV),* 2011 IEEE International Conference on, pp. 2423-2430. IEEE, 2011.

[7] D. Zhang, Y. Lu, L. Hu and H. Peng, 'Multi-human Tracking in Crowds Based on Head Detection and Energy Optimization', *Information Technology J.,* vol. 12, no. 8, pp. 1579-1585, 2013.

[8] I. Ali and M. Dailey, 'Multiple human tracking in high-density crowds', *Image and Vision Computing,* vol. 30, no. 12, pp. 966-977, 2012.

[9] F. Zhao and J. Li, 'Pedestrian Motion Tracking and Crowd Abnormal Behavior Detection Based on Intelligent Video Surveillance', *Journal of Networks,* vol. 9, no. 10, 2014.

[10] I. Ali and M. Dailey, 'Head Plane Estimation Improves Accuracy of Pedestrian Tracking in Dense Crowds', *Control Automation Robotics and Vision (ICARCV),* 2010 11th International Conference on. IEEE, 2010.

[11] D. Forsyth, 'Object Detection with Discriminatively Trained Part-Based Models', *Computer,* vol. 47, no. 2, pp. 6-7, 2014.

[12] S. Saxena, F. Brmond, M. Thonnat and R. Ma, 'Crowd Behavior Recognition for Video Surveillance', *Advanced Concepts For Intelligent Vision Systems,* vol. 9783540884576, p. 970, 2008.

[13] R. Eshel and Y. Moses, 'Tracking in a Dense Crowd Using Multiple Cameras', *Int J Comput Vis,* vol. 88, no. 1, pp. 129-143, 2009.

[14] A.B. Chan, Z.-S.J. Liang and N. Vasconcelos, 'Privacy preserving crowd monitoring: Counting people without people models or tracking', *Computer Vision and Pattern Recognition,* 2008. CVPR 2008. IEEE Conference on , pp.1,7, 23-28 June 2008

[15] V.K. Singh, Bo Wu and R. Nevatia, 'Pedestrian Tracking by Associating Tracklets using Detection Residuals,' *Motion and video Computing,* 2008. WMVC 2008. IEEE Workshop on, pp.1,8, 8-9 Jan. 2008

[16] M. Liem and D. Gavrila, 'Joint multi-person detection and tracking from overlapping cameras', *Computer Vision and Image Understanding,* vol. 128, pp. 36-50, 2014.

```
0 220
0 244
...
233 12
236 9

379 252
377 251
...
348 147
356 146
```

Fig. 5. Example output file. It shows that track 1 first appears at frame 0 and goes for 220 frames. Then track 2 appears at frame 0 as well but goes for 244 frames. After the new line in the middle of the file is where the coordinates begin. The first frame in track 1 is at position (379, 252) and the second frame is at (377, 251).

F. Tracking with Object Detection

G. Output

Using the information that we gathered from the above tracking methods we are able to create a output file with the track information. This output file is formatted into two sections.

The first section lists pairs of integers, with each pair representing a single track. The first integer states the starting frame (the frame in which the track first appears), and the second integer provides the number of frames for which the track appears.

The second section of the output file contains pairs of floats representing X and Z coordinates in a 3D environment. The pairs are listed in the same order that their respective tracks appear in the first section, and contain the coordinates for each frame of the tracks. An example output file is shown below:

[17] S. Ali and M. Shah, 'Floor Fields for Tracking in High Density Crowd Scenes', *Computer Vision ECCV 2008,* vol. 5303, pp. 1-14, 2008.

[18] S. Pellegrini, A. Ess, K. Schindler and L. van Gool, 'Youll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking', *Computer Vision,* 2009 IEEE 12th International Conference on, pp. 261 - 268, 2009.

[19] A. Yilmaz, O. Javed and M. Shah, 'Object tracking', *CSUR.,* vol. 38, no. 4, pp. 1-45, 2006.

[20] M. Thida, Y. Leng Yong, P. Climent-Prez, H. Eng and P. Remagnino, 'A Literature Review on Video Analytics of Crowded Scenes', *Intelligent Multimedia Surveillance,* pp. 17-36, 2013.

[21] R. Hartley and A. Zisserman, Multiple View Geometry, *Cambridge University Publishers,* 2nd ed. 2004

[22] G. Shu, A. Dehghan, O. Oreifej, E. Hand and M. Shah, Part-based Multiple-Person Tracking with Partial Occlusion Handling, *Computer Vision and Pattern Recognition,* 2012 IEEE Conference on (pp. 1815-1821), 2012.

[23] P. Viola and M. Jones, 'Rapid Object Detection using a Boosted Cascade of Simple Features Computer Vision and Pattern Recognition Proceedings', *2001 IEEE Computer Society Conference*, vol. 1: I-511 I-518, 2001.