Cluster-based Crowd Movement Behavior Detection

Meng Yang1, Lida Rashidi1, Aravinda S. Rao 2, Sutharshan Rajasegarar3, Mohadeseh Ganji1, Marimuthu Palaniswami2, Christopher Leckie1

1School of Computing and Information Systems, The University of Melbourne, Australia.
2Dept. of Electrical and Electronic Eng., The University of Melbourne, Australia.
3School of IT, Deakin University, Geelong, Australia.

Email: {myang3@student., rashidil, aravinda.rao, palani, caleckie}@unimelb.edu.au, srajas@deakin.edu.au, mohadeseh.ganji@gmail.com

Abstract—Crowd behaviour monitoring and prediction is an important research topic in video surveillance that has gained increasing attention. In this paper, we propose a novel architecture for crowd event detection, which comprises methods for object detection, clustering of various groups of objects, characterizing the movement patterns of the various groups of objects, detecting group events, and finding the change point of group events. In our proposed framework, we use clusters to represent the groups of objects/people present in the scene. We then extract the movement patterns of the various groups of objects over the video sequence to detect movement patterns. We define several crowd events and propose a methodology to detect the change point of the group events over time. We evaluated our scheme using 6 video sequences from benchmark datasets, which include events such as walking, running, local merging, global merging, global splitting and local splitting. We compared our scheme with state of the art methods and showed the superiority of our method in accurately detecting the crowd behavioral changes.

Index Terms—crowd behavior, event change detection, fixed-width clustering, video surveillance

1. INTRODUCTION

In recent years, crowd behavior monitoring and prediction has become an important topic in video surveillance. Although people typically behave in ordinary ways, an occurrence of a particular event can cause a panic. Therefore, ensuring the security and safety of the individuals in public places such as crowded streets, sports events, or shopping malls in cases of emergencies (terrorism, collapse, fire) warrants attention. The aim of this paper is to provide an automated video analytics approach for crowd event detection.

An important objective of crowd monitoring is to provide a logical model of people movements in case of an unexpected event, i.e., path planning of evacuation. Researchers [2] have shown the importance of automatic analysis for crowd behavior and its applications, such as crowd management, which is about strategies for building evacuation in case of disaster, and alerting authorities when detecting unexpected events captured through video surveillance footage. Crowd monitoring poses many challenges due to varying environments, such as irregular illumination conditions, shadows, non-rigid pedestrian movement and occlusions [4]. There have been several existing approaches applied to video surveillance analytics, however crowd behaviour understanding is still a challenge. Many methods are based on object detection and tracking [3], but tracking multiple individuals in crowded scenarios is a very challenging task. Therefore, in this paper, we propose a scalable method that can be used for crowd movement change detection, like sudden running, merging and splitting.

![Fig. 1. Overview of our crowd monitoring system, which includes video/image input (Step1), detection and clustering (Step2), event classification (Step3) and crowd event change detection (Step4).](image)

![Fig. 2. Sample frames of crowd events from PETS2009 dataset. (a) Walking: people walk across the scene from right bottom to left top. (b) Running: people start running at a specific frame. (c) Merging: people from different directions merge at the center of the scene. (d) Splitting: a crowd of people split into three different small groups. (e) Local Merging: two groups of people merge at first, and then walk to the left border as one group. (g) Local Splitting: a group of people moves toward the center of screen, then splits into two smaller groups.](image)
for analysis. The clusters are then used to detect abnormal behaviour, i.e., crowd change detection. In this paper, we used a fixed-width clustering algorithm [13, 14] to cluster the people into several groups (clusters) who are close to each other. The cluster centroids are used as the representative of each group of people. We then use the movement behavior of the clusters, captured via features such as the number of clusters, the distances between the centroid of the clusters and the velocity of change in cluster centroid positions, to detect various crowd event changes, such as merging, splitting and running events.

A brief overview of our proposed real-time event detection framework is shown in Fig. 1, where we first use YOLOv2 [11] for object detection in each frame, and the obtained objects are represented as the feature vectors at a spatial level. We then apply our proposed fixed-width clustering algorithm to find the group of objects in the frame, and then use the adjacent frames of the video sequence for event detection. The events that we focus on can be classified into two main categories, walking/running and merging/splitting. The merging and splitting event is further classified into local and global events. Each video sequence is then converted into a time series for crowd event change detection. Our framework is capable of detecting events such as walking, running, crowd formation (merging), scattering/splitting and local merging detection, as shown in Fig. 2. The main contributions of this paper are as follows:

1) Our proposed method is real-time since it can detect the events as soon as they occur. The framework makes use of only the detected objects coordinates for event detection, which is often in contrast to other existing algorithms that use shape, appearance and even audio information.

2) We use a novel cluster based method to represent groups of objects and define events by analysing the cluster movement profiles in a video sequence.

3) We conduct experiments on six video sequences of PETS2009, which contain global events: walking, running, merging, splitting and local events: partly merging and splitting. The evaluation reveals that such events can be detected, and the result is competitive when compared with other existing state-of-the-art algorithms.

The quantitative accuracy for all six sequences are all more than 80%.

The rest of the paper is organized as follows. Section II reviews previous algorithms for video-based crowd event detection, and Section III presents the problem statement. Further, our proposed methodology is introduced in Section IV, and Section V shows the experimental results on six sequences from PETS2009 at a global level and local level. Finally, the conclusion is given in Section VI.

II. RELATED WORK

There are several existing methods for crowd behavior analysis. In this section, we review the approaches for anomalous event change detection, which are more related to the aim of this paper. In order to address this task, spatio-temporal information based methods are quite popular, and these approaches can be grouped into two types: trajectory analysis and motion representation. Trajectory analysis based algorithms use tracking to obtain tracks and then distinguish abnormal ones from normal tracks in the scenario [5]. Motion representation methods use flow analysis to monitor events, for instance, the architecture in [18] uses Lagrangian particle dynamics with chaotic modeling to monitor and localize the abnormal objects and activities in crowded scenes. Further, representative tracks are identified and used as the compact modeling features. These trajectories produce temporal data that can be utilized for chaotic modeling in a simple way. The approach in [1] builds an optical flow related framework based on the stability of a dynamical model that can detect pre-defined events in the crowd. Moreover, Su et al. [6] used the spatio-temporal volume for variation matrix calculation, whose corresponding eigenvalues are obtained to describe local fluctuations.

Another class of approaches investigate theoretical models of crowd dynamics for event analysis. In [17], a social force model (SFM) is explored for anomalous activity detection and localization. The authors used the interaction of particles from optical flow information, which is estimated using the SFM, and these features are used for event analysis. Chen et al. [19] used optical flow based clustering for human crowd grouping in an unsupervised manner, and this method is called adjacency-matrix-based clustering (AMC). Zhou et al. [7] provided a novel mixture model of dynamic human agents (MDAs) that could learn the collective behaviour of people. The detailed approach is based on a bag of trajectories for MDA learning, then measuring the likelihoods of the tracks with the MDAs for anomalous behaviour detection. In [20], a Bayesian model is proposed for detecting escape events in crowded scenarios. First, foreground pixels and optical flows are extracted and used for crowd behavior characterisation, which can model escape and non-escape crowd behaviour using a Bayesian model.

As discussed above, most crowd event detection strategies in this domain rely on either tracking algorithms or complex modeling techniques. For instance, the authors of [5] used a tracking method to follow individuals in a crowded scene. However, it suffers from high computational complexity, and the accuracy of tracking based algorithms is low, because of overlapping and occlusion. In contrast, modelling-based strategies have been proposed, such as SFM [17]. Although such methods can achieve accurate crowd event detection, the number of parameters to be tuned is very large. Tuning such parameters relies on domain knowledge or pre-processing techniques.

III. PROBLEM STATEMENT

In order to achieve real-time crowd event detection, our proposed scheme utilises an unsupervised clustering method to characterize human behaviour. The groups of people in a scene are clustered, and the evolution of the clusters are used
to detect the change in crowd behavior. A brief overview of our proposed monitoring system is shown in Fig. 1. Step1: the video/image sequences are provided as input. Step2: The objects/people present in the image are detected and the groups of objects are found using a clustering method. Step3: Various events of crowd activity are detected, such as walking/running and merging/splitting; and Step4: changes in the crowd events are detected using a novel change detection method.

Consider an input as a video sequence $Vid = \{f_i, i = 1, ..., t\}$, where $f_i$ is the frame number. The aim is to find the time when an event change happens $P(f_{t+1}'), \ t' = 1, ..., t$. In each frame $f_i$, we denote the detected objects as $S = \{s_j, j = 1, ..., n\}$, where $n$ is the number of detected objects. For each pedestrian, we take the centroid of the detected bounding box as the coordinates of this object, which is described using $o_i = (x_i, y_i)$. Followed by being represented as a spatial feature $M_i = \{m_{ik}, k = 1, ..., n_b\}$, where $n_b$ is the block size. We used block sizes of 64 (i.e., the frame is divided into 8 by 8 blocks), 256 (16 by 16) and 1024 (32 by 32). The feature matrix of all the incoming frames $M = \bigcup_{i=1}^{n} M_i$, is used for clustering, using a fixed-width clustering algorithm.

The PETS2009 dataset [9] is used for evaluation in this paper. The six different types of crowd events that we analyze are as defined in [10]. The process of our proposed architecture is summarized in Fig. 3. After obtaining the clusters, the events are detected. They are categorized into two main types: Walking/Running and Merging/Splitting. Then, in the later stage, these events are further classified into four sub-types: Global Merging, Global Splitting, Local Merging, and Local Splitting. Finally, each video sequence is converted to a time series for crowd event change detection. The final output provides the time instance of a particular $P(f_{t+1}')$.

IV. EVENT DEFINITION

A. Walking and Running Event

1) Walking: This is a human behavior where a group of pedestrians move at a normal (average) velocity, which is much slower than the velocity of running events.

2) Running: This event is associated with the above walking event, but with a faster group movement.

Definition1: In order to detect a (group) event, we consider two adjacent video frames in the video sequence. In order to track the movement of a particular cluster (a group of objects/people) from the first frame to the second frame, we need to find the association of the same cluster in both frames. We find this associated cluster in the second frame using the nearest cluster of each cluster in the adjacent frame (step3). Finally, cluster features are used to produce the time series event curve and detects any change in the curve (step4).

B. Merging and Splitting Events

1) Global Merging: Pedestrians from different directions converging to the same point in the scene. This is denoted as $M = \{M_G\}$, where $M_G$ is for global crowd merging.

2) Global Splitting: This is the opposite event compared with merging, where people from the same crowd move to different points of the scene, and is denoted as $S_G$.

3) Local Merging: This event is a more complicated activity compared with global group merging. In some sequences of PETS2009, it is observed that some pedestrians merge at first, and walk as one group, then others come to merge. In this case, we use $M_L$ to represent this event, so the merging event can be denoted as $M = \{M_G, M_L\}$.

4) Local Splitting: Similar with local merging, a local splitting event is shown as a group of people who move toward the center of the screen, then a small part of this group split into another group, the rest of the original group walk until they split at another point. We denote this event as $S_L$, so the whole splitting event is denoted as $S = \{S_G, S_L\}$. 

Fig. 3. The steps involved in the proposed crowd event detection approach. As the video sequence is presented (step1), the pedestrians are detected and clustered (step2), and then the numbers and direction of the center of each cluster is recorded and used for event classification. The blue nodes denote the nearest cluster of each cluster in the adjacent frame (step3). Finally, cluster features are used to produce the time series event curve and detects any change in the curve (step4).
Algorithm 1 Event Change Detection System

Require: Video Sequence Vid = {fi, i = 1, ..., t}
1: while i ≤ t do
2: YOLO Detection : For each detected pedestrian S = {sj, j = 1, ..., n}, the recorded coordinate is oj = (xj, yj).
3: Fixed - width Clustering : Representing vectors vj and clusters C = {cj : r = 1, ..., Ne}.
4: Event Classification :
   - Walking/Running ← Definition1
   - Merging/Splitting ← Definition2
5: end while
Ensure: W/R or M = {MG, ML}/S = {SG, SL}
6: while 2 < i < t do
7: YOLO Detection,
8: Fixed - width Clustering.
9: Event Change Detection :
10: if W/R then
11: y - axis value ← calculate the velocity of C
   Event change point ← sharply rise/decline
12: else
13: if MG/SG then
14: y - axis value ← calculate the distance of C
15: Event change point ← sharply rise/decline
16: else
17: Locally merge ← CN of Gi decreases to 0
18: Locally split ← CN of Gi increases
19: end if
20: end if
21: i = i + 1
22: end while
Ensure: Event detected P(i′+1)

Definition2: Similar to Definition1, we consider two adjacent video frames in the video sequence. We find the associated cluster in the second frame using the nearest cluster principle, i.e., we find the nearest cluster in the second frame, and treat that as the associated cluster of the first frame’s corresponding cluster. We then record the change in direction, CD, of the center of each cluster, and use this to define events based on the following criteria: we randomly take one cluster’s direction CD and calculate the angles between the more distant cluster’s direction with itself. If there is more than one angle that is larger than 90°, this event is defined as a M/S event.

Definition1 and Definition2 are used for classifying W/R and M/S events, which are based on the first two adjacent frames of a video sequence. Definition3 and Definition4 that we present below are used to find the event occurrence time of global merging and local merging, which are based on the whole video sequence. The number of clusters CN should be considered along with the cluster direction change CD, which are described below in Definition3 and Definition4.

Definition3: We randomly take one cluster, and record the number CN and the direction CD of the clusters that are more distant from its surrounding ones. These more distant clusters could be identified as different groups G = {Gi, i = 1, ..., ng}, where ng represents the number of groups. If all of the distant clusters in G decrease to 0, it means all groups of people have merged together. The time/frame is identified as the event occurrence point for global merging.

Definition4: We randomly take one cluster, and record the number and the direction of the clusters that are more distant from its surrounding ones. These more distant clusters can be considered as different groups G = {Gij, k = 1, ..., i, j, k = 1, ..., ng}, where ng is the number of groups. If the number of clusters CN for any one group in G, like Gk decreases to 0, it indicates that two groups of people have merged together. The time/frame is identified as the event occurrence point for local merging.

V. METHODOLOGY

In this section, we describe the various components of our proposed framework in detail. A brief overview of our proposed monitoring system is shown in Fig. 1, where we first use YOLOv2 [11] for object detection in each frame, and the detected objects are clustered using our proposed fixed-width clustering approach. We then use each pair of adjacent frames in the video sequence and Definition1 for the event definition. In order to detect the W/R event, we consider the time series curve of velocity of the centroid of each cluster, and detect any sharp rise or fall, i.e., change of the curve. An example curve is shown in Fig. 3 that shows a W/R event detected using change detection of the velocity curve. For the M/S event type, based on Definition3, when the video sequence is identified as MG/SG, we calculate the Euclidean distance of each cluster and take the average value as the y-axis value of the curve. Like the sample curve shown in Fig. 3, our system can do MG/SG event change detection and event monitoring. About the ML/SL type, which is automatically classified using Definition4, if CN for one Gi decreases to 0, it means two groups of people merged at first. The whole process is summarized in Algorithm 1.

A. YOLO Object Detection

YOLO is a state-of-the-art, real-time object detection system. In this paper, we use YOLOv2 as our object detection tool [11]. In terms of many existing detection systems, they repurpose classifiers for detection, and apply the architecture to each frame at multiple scales and locations. Then the high scoring regions are recognized and used for detection.

In the YOLO scheme, it divides the image into an S × S grid. For each grid cell, it predicts B bounding boxes and confidence scores. In each grid cell it also predicts C conditional class probabilities that could be described as Pr(Class, Object). The C are conditional on the grid cell that contains the object. During the testing phase, the class-specific confidence score for every box is calculated using
Finally, the whole cluster set \( C \) \( \{c_i : i = 1 \ldots N\} \) is obtained, and it is used for crowd event change detection.

**B. Fixed-width Clustering**

In terms of the feature representation, we use an 8 by 8 block size. For example, if the detected object is at block 26, the feature is represented as a vector \([0, 0, \ldots, 1, \ldots, 0]\), where the value 1 occurs at the 26th position in the vector. Then, the obtained feature matrix \( M = \bigcup_{i=1 \ldots m} M_i \) is considered as the input for fixed-width clustering.

The purpose of the clustering [13, 14] is to group nearby people at the spatial level. The steps involved in this process are as follows: At first, a fixed-width clustering is used. The initial cluster \( c_1 \) is centred on the first vector \( M_1 = \{m_{1k}, k = 1 \ldots n_b\} \) with a fixed cluster radius \( \omega \). Then in general for each \( c_i \), the Euclidean distance \( d_i \) between the nearest vector \( c_r \) and itself is computed.

\[
d_i = \text{Distance}(c_i,c_r)
\]

If the distance \( d_i \) is less than the radius \( \omega \) of the nearest cluster \( c_r \), the vector will be added to that cluster, and the centroid \( m_r \) will be updated using the new vector. Otherwise, if it is larger than \( \omega \), this vector is used to form a new cluster \( c_f \). We repeat this process until all vectors are considered.

\[
\text{update}(c_r, m_r) = \begin{cases} 
c_r + 1, & \text{if } d_i \leq \omega \\
C \cup c_f, & \text{if } d_i > \omega 
\end{cases}
\]

After obtaining the clusters from the last step, for each pair of clusters \( c_r \) and \( c_j \), we compute the distance \( D_r \) between their cluster centroids. If \( D_r \) is smaller than the threshold \( \tau \), these two clusters \( c_r \) and \( c_j \) are merged to form a new cluster \( c_m \).

\[
c_m = \text{MergeCluster}(c_r, c_j), \quad \text{if } D_r \leq \tau
\]

Finally, the whole cluster set \( C = \{c_i : i = 1 \ldots N\} \) is obtained, and it is used for crowd event change detection.

**C. Event Classification**

Considering adjacent frames of the video sequence, for each cluster \( c_i^{t1} \) in Frame 1, the associated cluster in Frame 2 is selected as the nearest cluster \( c_i^{t2} \). The change in direction of each cluster is denoted as \( D_c = \text{Direction}(c_i^{t1}, c_i^{t2}) \), where \( i = 1 \ldots n, n \) is the number of clusters.

At first, we aim to classify these events into two main types: \( W/R \) and \( M/S \). If it satisfies the following equation, this event will be identified as \( W/R \).

\[
\text{Event} = \text{W/R}, \text{ if } \{\text{Ang}(C_{D_m}, C_{D_n}) \leq 90^\circ\}
\]

where \( m, n \in 1 \ldots n_p, n_p \geq 80\% \times n \), \( \text{Ang}() \) is the angle between two cluster directions.

Otherwise, i.e., if it does not satisfy equation (5), we then randomly take one cluster \( C_{D_m} \) and calculate the angles between the clusters \( C_{D_j} \) that are far away from its surrounding ones and itself. If there is more than one angle that is larger than \( 90^\circ \), i.e., \( \text{Ang}(C_{D_m}, C_{D_j}) \geq 90^\circ \), then this event will be classified as \( M/S \).

\[
\text{Event} = \text{M/S}, \text{ if } \{\text{Ang}(C_{D_m}, C_{D_n}) > 90^\circ\}
\]

where \( m, n \in 1 \ldots n_p, n_p \geq 2 \). After the above step, we still need to further classify if it is a global or local \( M/S \) event.

For detecting global/local merging, the number of clusters \( N \) is considered along with the cluster direction change \( C_{D_j} \).

To do that, we randomly take one cluster \( C_{D_m} \), and record the number \( C_{N_f} \) and the direction of the clusters \( C_{D_j} \) that are far away from its surrounding ones. Based on \( \text{Ang}(C_{D_m}, C_{D_j}) \), these far away clusters can be identified as belonging to different groups \( G = \{G_i, i = 1 \ldots n_g\} \), where \( n_g \) is the number of groups. If \( C_{N_f} \) of all \( G_i \) decreases to 0, it reveals that all the groups of people have merged together. This event could be identified as a global merging event \( M_G \).

On the other hand, if \( C_{N_f} \) of one \( G_i \) decreases to 0, it means that two groups of people are merged at first. This event could be identified as a local merging event \( M_L \).

We can detect both \( M/S \) based on the curve of the centroid distance, which we discuss in the following section.

**D. Crowd Event Change Detection**

In order to detect a Walking/Running event, we take the average coordinates \( \text{Aver}C = \{\text{Aver}C_i, i = 1 \ldots n\} \) of all clusters, where \( n \) is the total number of frames. We monitor the velocity change of \( \text{Aver}C \), and use velocity \( V_{W/R} = \{\text{Velo}(\text{Aver}C_i), i = 1 \ldots n\} \) as the y-axis value. The video is converted to a time series with the frame number as the x-axis and \( \text{Velo}(\text{Aver}C) \) as the y-axis. Our aim is to detect the specific frame number as the crowd event change point. When the \( \text{Velo}(\text{Aver}C) \) changes smoothly, that event is identified as a walking event. If the \( \text{Velo}(\text{Aver}C) \) changes sharply, then that event is recognized as a running event. The frame number at which the sudden change in \( \text{Velo}(\text{Aver}C) \) occurs is taken as the change point.

In terms of a \( M_G/S_G \) event, we use the average distance \( V_{M/S} = \{\text{AverDis}(C_m, C_k), i = 1 \ldots n\} \) of each pair of clusters as the y-axis value, where \( C_m \) and \( C_k \) are the clusters. The video will be converted to a time series with \( V_{M/S} \) as the y-axis and the frame number as the x-axis. When \( V_{M/S} \) is large, the event is identified as splitting, and after a sudden sharply decline, \( V_{M/S} \) changes to be a small value, which is recognized as merging. Otherwise, the y-axis value changes from small to large, and the change point is identified as the time when the change occurred.

For a \( M_L/S_L \) event, as with \( M_G/S_G \), we take the average distance \( V_{M/S} = \{\text{AverDis}(C_m, C_k), i = 1 \ldots n\} \) as the y-axis value. We randomly take one cluster \( C_i \), if the \( C_{N_f} \) of one \( G_i \) decreases to 0 at time \( t \), where \( i = 1, \ldots, l \), and \( G_i \) are the
groups that are not close to the group $C$, where $C_N$ is the size of $C$. This means only two groups merged, which is identified as local merging, and we note the time of this merge. On the other hand, if $C_N$ of one $G_i$ increases sharply, it means a local split has occurred.

VI. RESULT AND DISCUSSION

A. Experimental Setup

Our proposed crowd event detection monitoring system is implemented in Java. Object detection is based on YOLOv2, and we use the pre-trained darnet model directly, which used ordinary public datasets for training. Our system combines the YOLOv2 based detection with the object representation and fixed-width clustering. Finally, the resulting time series curve is obtained and analyzed. We use Windows 7 (64bit) with a 4 GB NVIDIA graphics card (NVIDIA NVS 315 HD), a multi-core Intel? i7-4790 3.6GHz CPU and 16GB RAM.

B. Datasets

The PETS2009 S3-High-Level (S3-HL) and S3-Multiple-Flow (S3-MF) datasets [9] are used for evaluating our proposed approach (refer to Table I).

The crowd events analysis are categorized from the dataset S3-HL and S3-MF dataset. The sequences that we choose are those where there is a change in the motion pattern (speed and/or orientation), such as people suddenly starting to run (change in speed), or merging/splitting from a point (change in direction). The ground truth values are the frame number when the change starts [15, 16].

We conduct experiments on the above video sequences and compare our results with existing state-of-the-art approaches that also perform event detection in such sequences. In the following section, we describe S3-HL, 14-16, View-001, part1 as Seq1, S3-HL, 14-16, View-001, part2 as Seq2, and S3-MF, 14-37, View-001 as Seq6.

C. Parameter Setting

The threshold setting for YOLO is set to 0. For fixed-width clustering, the radius of each cluster is set to be $\omega = 0.5$ for all six video sequences in the PETS2009 dataset. The merging threshold is $\tau = \omega / 2$.

In order to show that our results are not sensitive to these parameter settings, we conduct an experiment on Seq1 with varying values of cluster radius $\omega$. Fig. 4 shows how the detection accuracy varies with the chosen cluster width setting $\omega$. When $\omega \in [0.2, 3]$, the accuracy does not change, so we could pick any value within this range. For other cluster radius $\omega$, the accuracy is still quite competitive. We discuss how we obtain these accuracy values in the following subsections.

![Figure 4](image1.png)

**Fig. 4.** Parameter setting experiment on Seq1: x-axis is the cluster radius $\omega$, y-axis is the associated detection accuracy.

![Figure 5](image2.png)

**Fig. 5.** PETS2009 S3-HL, 14-33, View-001 (Seq3): Sample output time series obtained for merging/splitting event, x-axis is the frame number, y-axis is the average value of Euclidean distance between each pair of clusters. Red node denotes the detected event change point.

![Figure 6](image3.png)

**Fig. 6.** Seq3, merging/splitting event: (a) Frame detection, baseline with the ground truth, other state-of-the-art methods, and our detected frame marked in red. (b) Frame 1 of the video sequence. (c) Frame 333. (d) Frame 337.
D. Evaluation

We use the first four sequences for global detection and the latter two sequences for local detection. Fig. 5-11 shows the results of the proposed approach. The evaluation results show that they are all competitive when compared with the other methods.

![Evaluation Results]

For PETS2009, it is 7 frames per second, so our detected result is a 0.29s delay after the ground truth. For Fig. 7, the ground truth is Frame 38, and our detected result is Frame 41, so our detected result is a 0.43s delay after the ground truth. The same is observed with Fig. 8-9, it is 1.43s and 0.43 delay respectively. Furthermore, for these four videos sequences, our result is closest to the ground truth when compared with the other existing approaches.

In order to evaluate the result, we define the accuracy to be:

\[ \text{Accuracy} = 1 - \frac{\text{Total Frame}}{\text{Ground Truth}} \]

If the detected frame is closer to the ground truth, the performance is better. The quantitative evaluation is summarized in Table II.

In terms of local event detection, Seq5 shows a crowd splitting event where a group of people moves toward the center of the screen, then splits into two smaller groups (refer to Fig. 10). (a) is the ground truth when the local event first occurs, and the detection frame based on other existing methods and our method; (b) is a sample frame to show the whole event; (c) is the ground truth frame; and (d) is our detected frame; (e) is another sample frame to show the whole event. Our system detected the first splitting activity at frame 81, where the ground truth is at Frame 76, and Frame 83 is the result of the WSC method. We have a has 0.7s delay,
In this paper, we propose a crowd event change detection framework that can achieve good performance at both a global level and local level. The architecture of our framework is based on the YOLOv2 model for object detection and our proposed coordinates representation and the fixed-width clustering is used to produce the clustering result, like cluster coordinates, the numbers of clusters, and changes in cluster direction. Further, these features are used for event classification and event change detection. The events are classified into two main types, walking/running and merging (global and local)/splitting (global and local). Six video sequences of the PETS2009 dataset are used for evaluation. In comparison with other existing methods, our experimental results show that the accuracy is between 80%-95.7%. On all the video sequences our method achieves very competitive performance.

VII. CONCLUSIONS

In this paper, we propose a crowd event change detection framework that can achieve good performance at both a global level and local level. The architecture of our framework is based on the YOLOv2 model for object detection and our proposed coordinates representation and the fixed-width clustering is used to produce the clustering result, like cluster coordinates, the numbers of clusters, and changes in cluster direction. Further, these features are used for event classification and event change detection. The events are classified into two main types, walking/running and merging (global and local)/splitting (global and local). Six video sequences of the PETS2009 dataset are used for evaluation. In comparison with other existing methods, our experimental results show that the accuracy is between 80%-95.7%. On all the video sequences our method achieves very competitive performance.

REFERENCES