Crowd Activity Change Point Detection in Videos via Graph Stream Mining

Meng Yang[1,2], Lida Rashidi[1,2], Sutharshan Rajasegarar[3,4],
Christopher Leckie[1,2], Aravinda S. Rao[4], Marimuthu Palaniswami[4]

[1]School of Computing and Information Systems, The University of Melbourne
[2]Data61 Victoria Research Laboratory
[3]School of Information Technology, Deakin University
[4]Department of Electrical and Electronic Engineering, The University of Melbourne

{myang3,lrashidi}@student.unimelb.edu.au
sutharshan.rajasegarar@deakin.edu.au
{caleckie,aravinda.rao,palani}@unimelb.edu.au

Abstract

In recent years, there has been a growing interest in detecting anomalous behavioral patterns in video. In this work, we address this task by proposing a novel activity change point detection method to identify crowd movement anomalies for video surveillance. In our proposed novel framework, a hyperspherical clustering algorithm is utilized for the automatic identification of interesting regions, then the density of pedestrian flows between every pair of interesting regions over consecutive time intervals is monitored and represented as a sequence of adjacency matrices where the direction and density of flows are captured through a directed graph. Finally, we use graph edit distance as well as a cumulative sum test to detect change points in the graph sequence. We conduct experiments on four real-world video datasets: Dublin, New Orleans, Abbey Road and MCG Datasets. We observe that our proposed approach achieves a high F-measure, i.e., in the range [0.7, 1], for these datasets. The evaluation reveals that our proposed method can successfully detect the change points in all datasets at both global and local levels. Our results also demonstrate the efficiency and effectiveness of our proposed algorithm for change point detection and segmentation tasks.

1. Introduction

A major challenge in video surveillance is detecting unusual patterns of activities, i.e., anomalous event detection. Anomalous events correspond to time points in a video that are different from the frequently occurring patterns at a spatial, temporal or spatio-temporal level. Applications of this type of data analysis are detecting irregular crowd/individual movement/activities.

Detecting anomalous crowd activity is a challenging analysis problem, particularly since crowds are collections of individuals that are identifiable when the density of people is high enough to disrupt individual and group identification [1]. Crowd abnormality detection is further complicated by frequent occlusions, low quality video, and ambiguous definitions of anomalous behaviors. Due to the complex nature of video streams, it is challenging to detect a particular incident/event that has occurred in a video sequence, and then distinguish the irregular events from normal patterns based on predefined rules. Therefore, we aim to detect irregular incidents in videos from a statistical perspective, in an unsupervised manner. In particular, we address the problem of detecting the timestamps when the crowd flow density changes dramatically between specific locations in a public environment.

This task is referred to as change point detection of flows. In our proposed framework, a real-time multi-object tracking algorithm is used for obtaining trajectories as the initial step. Next, similar trajectories are grouped together using a hyperspherical clustering method. The start and end locations of each cluster are recognized as interesting regions automatically. Further, we monitor the density of pedestrian flows between every pair of interesting regions over consecutive time intervals. The results of monitoring are then represented as a time sequence of the adjacency matrices, where the crowd flow is captured using directed graphs. In other words, every interesting region is represented as one node, and the flows of pedestrians between every pair of nodes are used as the edges of a directed graph. Finally, we use graph mining to detect change points of the flows. Our proposed framework is shown in Figure 1.

Traditional change point detection techniques analyze the video data either in its original pixel-by-pixel format,

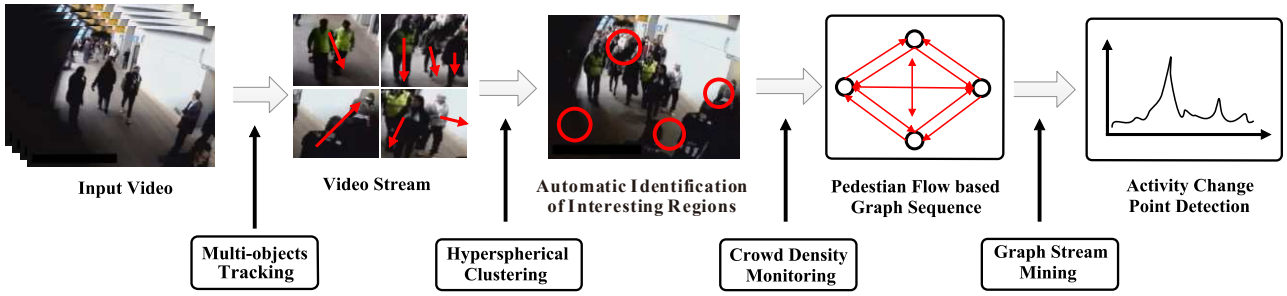

Figure 1: Overview of our framework.

which yields sparse crowd densities and high computational cost, or use a simple counting technique, which disregards the flow and direction of the crowd movement. The advantage of using a graph representation for a video stream is that it provides a trade-off between preserving crowd flow and direction information, in addition to a more computationally efficient summarization of the data. In particular, it enables the detection of localized structural changes in the flow pattern within large crowds.

The main contributions of this paper are as follows:

(1) We propose a novel graph mining framework to address the video surveillance problem. In particular we propose a method to detect anomalous directional flows of objects, such as pedestrians, between different regions of the monitored area in an unsupervised manner.

(2) A hyperspherical clustering method is used for automatically defining the interesting regions of the video scene. Therefore, regions like entrances, exits and bathrooms are recognized as interesting nodes automatically, which can be used at the next stage.

(3) We utilize an unsupervised technique for real-time multi-object tracking of pedestrian movements for crowd tracking. This removes the requirement for pre-labelled video sequences in the analysis.

(4) We conduct experiments on four real-world video datasets: Dublin, New Orleans, Abbey Road and MCG Cameras. The F_1 score varies between [0.7, 1] for these datasets. The evaluation reveals that our proposed method can successfully detect the change points in all datasets.

The rest of the paper is organized as follows. In Section 2, we describe the related work. Next, we present the problem statement and show our approach in Section 3. Further, the proposed video stream analytics method, which includes defining interesting regions automatically and crowd density monitoring are described in Section 4. In Section 5, we describe two series of experiments. First, we empirically demonstrate the effectiveness of our approach in detecting changes in directed crowd flows, which cannot be detected by conventional approaches that simply count the total number of pedestrians. Second, we evaluate the accuracy of activity change point detection using four benchmark video datasets. Finally, we conclude in Section 6.

2. Related Work

There are a number of existing approaches that have been applied for object detection and monitoring in image sequences and videos. In [2], the authors proposed a background subtraction method for extracting the pixels of foreground people at first, followed by mapping the number of pedestrians to perform counting. The authors in [3] show that the relationship between the number of people and extracted foreground pixels is approximately linear. However, the problem of this algorithm is its performance, which deteriorates due to occlusion and perspective effects. It is difficult to follow and link each person from one frame to the next frame by just using foreground pixels. Therefore, a deep learning method faster R-NN is used for object detection [4]. In [5], crowd features are described by a maximally stable extremal region (MSER). In [6], Kanade-Lucas-Tomasi (KLT) based corner points are extracted as features, followed by clustering these features for human detection. Rittscher et al. [7] proposed a contour feature based approach, which groups human-sized models by utilizing Expectation Maximization (EM).

Simply doing object counting and tracking cannot achieve our anomaly detection requirements. There is some literature about abnormal crowd behavior detection, like using social force models [8] and a mixture of dynamic textures (MDT) for crowd behavior representation [9]. However, they focus on the anomalous individual detection, whereas we aim at doing abnormal activity monitoring and segmentation for a whole video.

3. Problem Statement

Given a surveillance video sequence, our aim is to find any anomalous crowd movement patterns between the interesting regions within the monitored area over time. This task requires several challenges to be addressed,

namely (1) measuring the directional flows of the crowd or pedestrians between the interesting regions, and (2) characterizing the normal flows and detecting the anomalous movement patterns in an unsupervised manner. In this work, we address these challenges by proposing a graph based change point detection method to detect anomalous directional flows of crowd movements in videos.

Our aim is to identify the times (timestamps) when a significant change in crowd movements has occurred between the nodes (regions), i.e., we want to find when the anomalous crowd movements have occurred over the observed time period. To identify such timestamps, we propose the following framework as shown in Figure 1.

The proposed framework comprises three main components: automatically defining the interesting regions, crowd density monitoring, and graph stream mining. First, we use a hyperspherical clustering algorithm [14] to group similar trajectories (of objects), and then extract the average trajectory for each cluster. The start and end locations of each average trajectory are recognized as interesting regions, which are used as nodes in each graph. Next, we monitor the crowd density between each pair of nodes, and the number of pedestrians moving from one node to another are used as the edge weight in each directed graph. The multi-object tracking algorithm uses ViBe [16] for foreground subtraction and moving object tracking. Thereafter, the pedestrians are tracked sequentially using a Kalman filter [17]. Then a Hungarian assignment algorithm [18] with occlusion handling is used to improve the performance of multi-object tracking. The output of this algorithm is the crowd density measurement between each pair of regions considered in a scene. This crowd density information between each pair of regions (nodes) is used to construct a series of weighted graphs, which are represented using graph adjacency matrices. Finally, we employ Graph Edit Distance (GED) [13] with a single-variable Cumulative Sum (CUSUM) test [10] for segmenting the graph data streams to detect the anomalous crowd movement instances over the monitored time period.

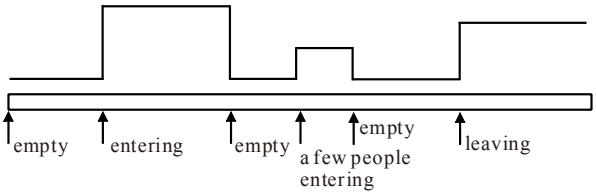

Figure 2: An example of crowd activity segmentation in the MCG stadium.

Once the movement patterns are represented by the graphs, we can use the graph mining method to find the change points of crowd flows in a video, which can be further used to segment the video at a temporal level. For example, Figure 2 shows the result of crowd activity segmentation performed on the graph, which reveals the sequence of activities over an extended time period. In this figure, the sequence of different activities observed is empty $\rightarrow$ people entering $\rightarrow$ empty $\rightarrow$ few people go across to the bathroom $\rightarrow$ empty $\rightarrow$ people leaving.

We now present each of the modules in our proposed framework in detail.

4. Video Stream Analytics

In this section, we first propose a hyperspherical clustering algorithm for automatically identifying interesting regions in an unsupervised manner. Next, we present the object detection and multi-object tracking approaches that are used for determining the crowd density between a pair of nodes/regions. This has several subtasks, such as video preprocessing to filter noise in the video, object detection and multi-object tracking for effective computation of the crowd movement density.

Algorithm 1: Automatic Identification of Interesting Regions

Require : obtained trajectory $T = \{t_i : i = 1 \ldots n\}$, cluster width ω and merging threshold τ
  $C \Leftarrow \varnothing$ { empty cluster set C }
  $N_C \Leftarrow 0$ { number of clusters }
  Create the first cluster c_i with the centroid t_i
  $C \Leftarrow c_1$
  $N_c \Leftarrow 1$
  for $i = 2$ to n **do**
    Compute the distance d_i between t_i and the nearest cluster $c_r \in C$ with centroid m_r
    if $d_i \leq \omega$
      $c_r \Leftarrow t_i$ {be added to c_r}
      $m_r \Leftarrow$ update the centroid
    else
      Create a new cluster c_f with the same centroid as t_i
      $C \Leftarrow C \cup c_f$
      $N_c \Leftarrow N_c + 1$
    end
  end
  Output : cluster set $C = \{c_r : r = 1 \ldots N_c\}$
  $C_m \Leftarrow \varnothing$ {empty merged cluster set C_m}
  $C_m \Leftarrow C$ {start with all original clusters}
  for $r = 1$ to N_c **do**
    for $j = r + 1$ to N_c **do**
      compute distance D_r *between* c_r *and* c_j
      if $D_r \leq \tau$
        $c_m = \text{MergeCluster}(c_r, c_j)$
      end
    end
  end
  Output : merged cluster set $C_m = \{c_m : m = 1 \ldots l\}$ with average trajectory $T_a = \{t_j : j = 1 \ldots l\}$ for each merged cluster c_m

4.1. Automatic Identification

A hyperspherical clustering algorithm [14, 15] is used to group similar trajectories, and produce the average trajectory for each cluster. Then, the start and end locations of every average track are used to identify interesting regions, which are represented as the nodes in each graph. The trajectories of moving pedestrians are obtained by using a real-time tracking algorithm, which is also used for crowd density monitoring in Section 4.2. In this subsection, we present how we employ a hyperspherical clustering algorithm to automatically identify the interesting regions in the scene.

The approach has three main steps: Trajectory Representation, Fixed Width Clustering and Merging Clusters. The entire procedure is shown in Algorithm 1 and Figure 3.

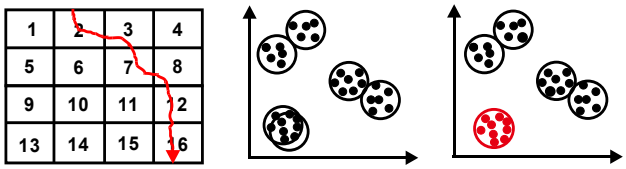

Figure 3: (a) a 4 × 4 blocks example. The track can be represented as the vector. (b) Fixed width clustering. (c) Cluster merging.

Trajectory Representation The approach that we used for representing the trajectories is proposed in [15]. Let m be the width and n be the height of a video frame. The input frame is divided into $b_x \times b_y$ blocks, where $1 < b_x \leq$ m, $1 < b_y \leq$ n. Block size selection affects the encoding results. In this experiment, we choose 16×16 as the block size. Next, a feature representation with spatio-temporal information is extracted for each object.

When an object enters a block, this block is assigned the original value 1. If this object stays in the same block, the value will be the number of frames that the object stays in that block. Further, if the object moves to another block, the value of this new block will start from the value of the former block plus one. The 4 × 4 block size example is demonstrated in Figure 3.a, the red track shown there can be represented as a vector [0 1 2 0 0 0 3 4 0 0 0 5 0 0 0 6] with index 1 to 16. All the tracks are represented as vectors of integers.

Fixed Width Clustering The first cluster is centred on the first trajectory, and has a fixed cluster radius. Next, the Euclidean distance is calculated between the next trajectory vector and the centroid of each cluster. If the Euclidean distance is smaller than the radius (width) of the nearest cluster, the track vector will be added to that cluster. The centroid of this cluster will be updated as the mean value of all vectors in it. On the other hand, if the distance is larger than the radius/width, the trajectory vector forms a new cluster with a new centroid. This process is repeated until all the trajectory vectors have been considered (see Figure 3.b).

Merging Clusters Based on the clusters we obtained in the previous step, we calculate the Euclidean distance between the centroids of each pair of clusters. When the distance is smaller than a threshold τ, the two clusters are merged into a new cluster. We repeat this process until all clusters have been considered (see Figure 3.c). Finally, the start and end locations of each cluster are recognized as interesting regions, which are represented as nodes in each graph.

4.2. Crowd Density Monitoring

Video Pre-processing and Object Detection First, the video sequence is converted to a grayscale image sequence, followed by the application of a 2D Gaussian low-pass filter, which eliminates the high frequency noise. The parameters of the Gaussian filter are set as $\sigma = 0.5$ and the block size = 5×5. These parameters are chosen in such a way that the filter retains the low-frequency and edge information in the output.

The object detection algorithms currently available in the literature can be categorized into three types: (1) optical flow (KLT), (2) differentiating consecutive frames, and (3) foreground or background subtraction. The third approach is a widely used technique for detecting moving objects and pedestrians in practice.

These techniques can be further categorized into parametric and sample-based methods, which consider the pixel changes of objects or crowds. However the latter approach is an improvement over the former one. The parametric method first builds an estimate of a Probability Density Function (PDF) for each pixel location, and then uses this to classify a pixel value as a background or foreground pixel value depending on how it fits within the estimated PDF. Considering the complexity of real-world scenarios, we propose to use a sample-based algorithm, called ViBe [13], for foreground subtraction. This approach is based on using a set of sample values to build the pixel model by aggregating previously observed values for each pixel location. Next, the model is updated regularly based on the incoming frames, and then, the moving pedestrians are recognized by subtracting the pixel model.

Multi-object Tracking Once the objects are detected, the next task is to track them. We use a Kalman filter [14] for object tracking in this work. The Kalman filter predicts the state (location, velocity) of an object from a sequence of

images. It finds the minimum mean-square error estimate of the state x_k sequentially using measurements and a transition model, where k denotes the time. To detect crowd movements, we need to perform multi-object tracking in the video. We use the Hungarian assignment algorithm [18, 19] in this paper. The Hungarian algorithm improves the accuracy of the track prediction of an object in the incoming frames by addressing the issue of multi-object link assignment.

In this approach, we first need to assign an index to every vision object that is detected by our tracking algorithm in the current frame, i.e., the detected positions. The set $Ind = \{ind_i : i = 1...I\}$ denotes the indices of objects, where I is the maximum number of objects in the current frame. These I vision objects need to be mapped to the I_w world objects, i.e., the predicted positions, optimally. The Hungarian assignment algorithm can solve this optimization problem in polynomial time. This algorithm builds an initial cost matrix C based on the Euclidean distances between the predicted and detected positions for each frame. This method results in an optimum object assignment matrix X, where all elements of each row are zero except for one. The optimization can be summarized as a minimization of the objective function shown in equation (1).

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

$$\text{where, } x_{ij} = \begin{cases} 1 & \text{if } e_i \text{ is assigned to } m_j \\ 0 & \text{otherwise} \end{cases}$$

Occlusion Handling In our implementation, we use a depth model to handle the occlusion problem to find more accurate and clear tracks. The proposed idea is introduced in [20] and is based on determining the depth of the tracked objects. For example, if an object is moving from far away towards the camera, the depth will be changing from large to small. Therefore, when two objects occlude, the depth model of each object can be used to separate the bounding box, which can improve the performance of the Hungarian assignment algorithm.

The first step is to generate the probability density functions for the depth of the moving objects at each pixel from a set of training data. We use the first 10 frames of each moving object as the training data, so the occlusion handling will start after Frame 10.

The probability density functions are obtained using equation (2). For each image pixel φ, we get an observed depth histogram $Z_\varphi(D)$. If pixel φ meets an occlusion with a blob ω, histogram $Z_\varphi(D)$ is aggregated on the depth of the blob, which can be represented as D_ω. We repeat this process using the training data (first 10 frames) on each detected moving target.

$$Z_\varphi(D) = \sum_{\omega \in \Omega_\varphi} \delta(D - D_\omega) \tag{2}$$

where Ω_φ is the set of blobs that contain φ and $\delta(\cdot)$ is the Kronecker delta function:

$$\delta(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{3}$$

Next, an activity map A_φ is generated using a set of observed depth histograms $Z_\varphi(D)$:

$$A_\varphi = \sum_{D \in D_{\min}, D(i_\varphi)} z_\varphi(D) \tag{4}$$

When an occlusion occurs, the depth model of each object is used to assign an accurate position to these objects in the next frame, which can improve the accuracy and effectiveness of the multi-object tracking algorithm.

4.3. Anomaly Detection

Once the crowd density movements are obtained, the next step is to represent them in the form of adjacency matrices that will be used in our change point detection scheme.

In terms of detecting anomalous timestamps, a common method to solve this problem is by looking at the generated probability distributions of data between the past and present. If the two distributions differ significantly from one time point to another, this point will be recognized as a change point. In this work, the crowd density of each path between the pairs of nodes/locations is represented as the weight of an edge in a directed graph. We use Graph Edit Distance (GED) with a single variable CUSUM test to analyze this graph data stream.

Graph edit distance [13] is used to measure the distances between two graphs G_1 and G_2, where the main assumption is that G_1 can be transformed into G_2 using a finite number of consecutive graph edit operations. These edit operations are defined specifically based on the type of graph, i.e., weighted, directed, undirected or simple.

The definition of GED $d_{\gamma_{\min}}(G_1, G_2)$ between graphs G_1 and G_2 is shown in equation (5).

$$d_{\gamma_{\min}}(G_1, G_2) = \min_{\gamma \in \Gamma(G_1, G_2)} \sum_{e_i \in \gamma} c(e_i) \tag{5}$$

where $\Gamma(G_1, G_2)$ denotes the set of all complete edit paths that transform G_1 to G_2, and c is the cost of an edit.

Since the graphs in our problem consist of weighted directed edges between every pair of nodes, we can make the assumption that the graph is complete. Therefore, the GED between G_1 and G_2 can be computed as the difference between the graphs' adjacency matrices. After computing

the time series using GED, we can use a CUSUM test to detect the change points.

Cumulative sum (CUSUM) tests [10] monitor the mean of a process based on the time series of data that is generated from that process at any given period, such as minutes, hours, days or years. In this paper, the consecutive intervals that we use are based on the video length. We use three minutes as the interval value, but for smaller datasets, the interval is considered to be 20 or 30 seconds. A CUSUM test can be employed for detecting anomalous timestamps or change points by comparing an accumulative data measure computed between the past and the present data samples.

5. Results and Discussion

In this paper, we conduct two series of experiments. In experiment 1, we discuss the intuition behind using a graph-based representation for detecting change points in directed crowd flows, which is compared to the standard approach of simply counting the aggregated number of pedestrians in the whole scene. In experiment 2, we evaluate the performance of our change point detection schemes at both global and local levels in four real video surveillance datasets.

In this section, we describe our experimental setup and datasets in Section 5.1. Experiments 1 and 2 are shown in Sections 5.2 and 5.3, respectively. In Section 5.4, we discuss the computational complexity and performance of our approach.

5.1. Experimental Setup and Datasets

Experimental Setup The proposed crowd density estimation approach is implemented in OpenCV 3.0.0 with Visual Studio 2013. The automatic identification of interesting regions is implemented in Java. The graph mining and final quantitative evaluation are implemented in Matlab 2013b. The configuration of the machine used for the evaluation is as follows: 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.

Datasets In order to evaluate our framework, we conducted experiments on four real-life video datasets: (1) the Dublin Cam (www.earthcam.com/world/ireland/dublin), (2) the New Orleans Cam (www.earthcam.com/usa/louisiana/neworleans/bourbonstreet) and (3) Abbey Road Crossing Cam (www.abbeyroad.com/Crossing) from EarthCam, in addition to (4) the Melbourne Cricket Ground (MCG) (www.mcg.org.au), i.e., a major sports stadium in the city of Melbourne.

The description of these datasets is summarized in Table 1, which includes the average number of people per 10 minutes, the main characteristics, and camera view. We assume these cameras are calibrated and the video data can be used directly. Figure 5 shows the sample frames taken from these video datasets.

Camera	Average People/10min $\pm\sigma^2$	Main Characteristics	Camera View	Time Length
Dublin	287 ± 53	moving	wide	3 hours
New Orleans	330 ± 48	moving	wide	2 hours
Abbey Road	206 ± 44	moving /static	wide	2 hours
MCG	234 ± 67	moving	wide	2 hours

* σ^2 =Standard Deviation

Table 1. Video data used in experiment 2

Parameter Settings In the component for the automatic identification of interesting regions, , the parameter ω (cluster width) is set to be 36, 40, 20 and 10 for the Dublin, New Orleans, Abbey Road and MCG datasets, respectively. The merging threshold $\tau = \omega / 2$. In terms of the CUSUM parameter settings, the user defined parameters h and k are chosen to be 4 and 0.4 respectively for the first three datasets and 4 and 0.3 for the MCG datasets.

5.2. Demonstration of Flow-based Analysis

In experiment 1, we use the video data between 8am and 3pm on 26th Oct 2017 from the Dublin Cam. This experiment explains the intuition behind using graph based analysis instead of the aggregated number of pedestrians to detect change points.

Figure 4.a shows a time series of the aggregated number of people in the previously mentioned dataset, where the aggregate values are computed within every 3-minute non-overlapping window. We divide the time interval into two periods: morning and busy time. As shown in Figure 6.a, we randomly select two time stamps within the busy time, and differentiate them with green and red markers. We observe that the aggregated numbers of people in these two time stamps are similar. However, based on our framework, we found that the directions of the crowd flow in the frames corresponding to these two time stamps are substantially dissimilar (see Figure 4.b). The direction in addition to crowd density along different paths in a frame can easily be represented through our directed graph model (see Figure 4.c). We have used GED with CUSUM to determine whether our graph-based approach can differentiate these two points. Since GED takes the change in weight and direction of each edge into account, it can easily detect that a change has occurred and these two time stamps are dissimilar.

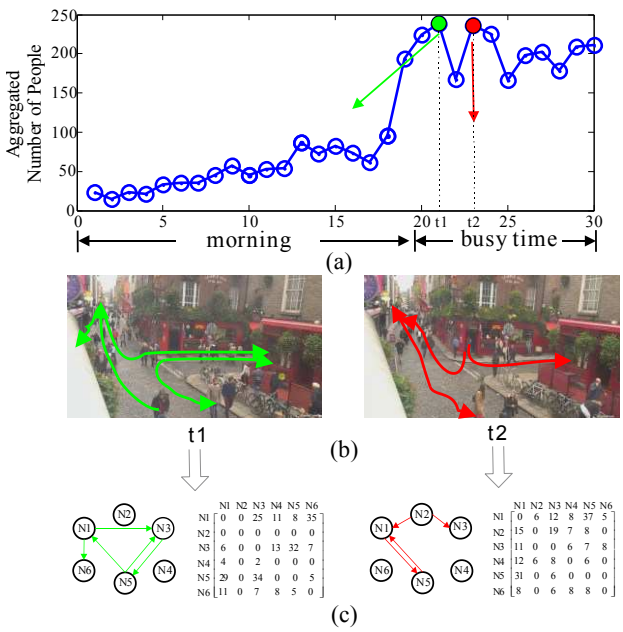

Figure 4: (a) Aggregated number of people, (b) graph pattern and (c) graph representation. (Dublin dataset for two intervals in busy period with similar aggregate numbers of people).

5.3. Activity Change Point Detection

In this section, we conduct experiment 2 on the four datasets as shown in Table 1 and Figure 5. The interesting regions have been automatically identified and denoted by red circles, where there are 6, 5, 5, and 4 interesting regions in Dublin Cam, New Orleans Cam, Abbey Road Cam and MCG C6 Cam, respectively. The pedestrian movement flows are converted to the graph representation (Figure 5).

We then construct our graph time series, where each graph is computed within a three minute non-overlapping window. Therefore, these videos can be represented as 60, 40, 40, 40 graph adjacency matrices for Dublin, New Orleans, Abbey Road and MCG respectively. Further, we conduct the following experiments on the previously mentioned datasets:

(1) Detecting change points at the global level: In terms of global-level change point detection, we aim to find the change points of the time series, i.e., time stamps where crowd density and direction underwent a major change in the entire graph.

(2) Finding change points at the local level: The local level change detection refers to analyzing the crowd density in single/multiple path(s) instead of considering an entire graph. We aim to segment the crowd activity for a given interesting region (node) using the crowd density on the paths connected to that node. Consider the Dublin Cam as an example. There are 6 nodes, so there are 30 paths including N12, N21, …, N65. We choose node 2 as the key node that we want to analyze, so we need to analyze every path that is connected to node2, such as N12 (node 1 to node 2), N21 (node 2 to node 1),…N26 (node 2 to node 6) and so on. After being analyzed at a local-level, we can segment the activity status of node 2. In the local-level based experiment, for Dublin Cam, node 2 is a bar or café, so we choose node 2 as the focus node. For the other three datasets, we just randomly choose node 1 as the focus node.

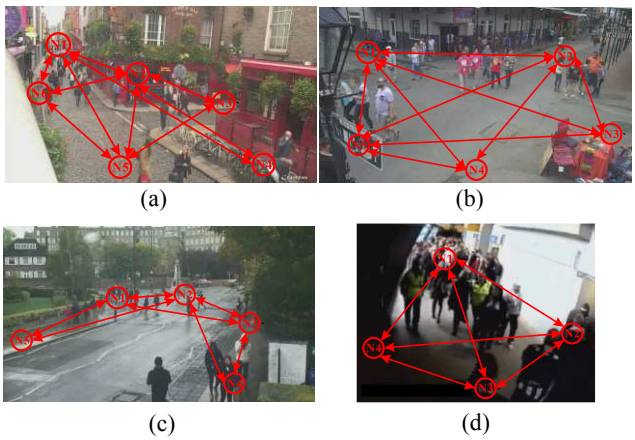

Figure 5: Graph representation of (a) Dublin Cam. (b) New Orleans Cam. (c) Abbey Road Cam. (d) MCG Cam.

We manually identified the change points in the four real datasets and used them as the ground truth. We then evaluate the results by using the F-measure [21, 22]. The confusion matrix that we used is illustrated in Table 2.

	Detected as change point	Detected as non-change point
True change point	TP	FN
True non-change point	FP	TN

Table 2. Confusion matrix for F-measure

F-measure or F_1 score provides a way to measure the overall effectiveness of a Change Point Detection (CPD) algorithm, which is based on the combination of precision and recall. F-measure is computed as the ratio value of the weighted importance of precision and recall. F_1 score $\frac{(1+\alpha)^2 \times \text{Recall} \times \text{Prediction}}{\alpha^2 \times \text{Recall} + \text{Prediction}}$, where Recall (R) and Precision (P) are computed using $\frac{\text{TP}}{\text{TP}+\text{FN}}$ and $\frac{\text{TP}}{\text{TP}+\text{FP}}$, respectively. In our experiment, we choose equal weights for precision and recall, so $\alpha = 0.5$. The F_1 score =

$\frac{2 \times \mathrm{R} \times \mathrm{P}}{\mathrm{R} + \mathrm{P}}$. F_1 score is in the range [0, 1], where higher values show better performance.

The evaluation of the global-level and local-level based CPD result is summarized in Table 3. We use the aggregate time series (as Experiment 1) based CPD with single variable CUSUM for evaluation as the baseline. The results are also summarized in Table 3.

As shown in Table 3, our global and local-level frameworks for CPD both outperform the aggregate time series based CPD. It shows the better performance using graph based analysis compared with the aggregated number of pedestrians to detect change points.

Video	Global-level	Local-level	Aggregate (baseline)
Dublin Cam	84.2%	81.5%	61.2%
New Orleans Cam	80.0%	75.0%	57.9%
Abbey Road Cam	80.0%	72.7%	57.4%
MCG Cam 6	92.3%	85.7%	64.3%

Table 3: F_1 score of the proposed approach on the four real datasets.

Another phenomenon in Table 3 is that the global-level detection has better performance than local-level detection. Further, we can see our method achieves the best performance on the MCG video, which is more than 90% at the global-level and 85% at the local-level, followed by the Dublin dataset, New Orleans and Abbey Road videos. The difference in performance is because of the complexity of the scene in some videos, such as more occlusion and overlapping. Generally, the results show that they can all achieve reasonable accuracy at a global-level and local-level.

5.4. Discussion

In this section, we discuss how our proposed framework can be used in real-time video streaming change point detection, in terms of computational complexity and the minimum number of frames required to detect an event.

Since our proposed framework is a hybrid of crowd density monitoring and graph mining, its computational complexity can be reported as $\mathrm{O}(n^4)$, in which the Kalman Filter is $\mathrm{O}(n^2)$ and the improved Hungarian algorithm is $\mathrm{O}(n^4)$, where n denotes the number of detected and assigned moving people in a frame. Our graph change point detection algorithm requires constant time a, since the number of nodes is a constant for the datasets. Therefore, the total computational complexity time should be $\mathrm{O}(n^2) + \mathrm{O}(n^4) + \alpha = \mathrm{O}(n^4)$, which is an efficient real-time algorithm compared with other tracking algorithms.

In terms of the required minimum number of frames to detect an event, the tracking algorithm merely requires 1-2 frames before the current frame to predict the velocity, direction and position of every object in the next frame. Therefore, the tracking can be performed in real-time, e.g., for real-time CCTV video streams. As for the change point detection algorithm, we require the video length at least to be 2 minutes (2760 frames at 23 fps) for the event detection to be effective.

Moreover, since our algorithm is capable of processing video data from video streams without needing supervised training data, it can be applied in scenarios where the streaming data is not labeled.

6. Conclusions

In this paper, we proposed a framework based on crowd density monitoring and graph mining for activity change point detection. First, a real-time multi-object tracking algorithm is used to obtain trajectories. Next, similar trajectories are grouped together using a hyperspherical clustering method. The start and end locations of each cluster are used to automatically identify interesting regions in the scene. Thereafter, we monitor the density of pedestrian flows between every pair of interesting regions over consecutive time intervals. The results of monitoring are then represented as a dynamic directed graph with adjacency matrices that capture crowd flow and direction. We then use GED with CUSUM to determine change points or anomalous incidents. We demonstrate the effectiveness of our approach in detecting localised changes in flows, in contrast to simply using aggregate pedestrian counts, as well as the accuracy of our approach on four real-world video surveillance datasets. The evaluation reveals that our proposed method can successfully detect the change points in all datasets. In addition to the effectiveness of our algorithm, it is able to produce interpretable results, which can explain change points at both the global and local levels. As a future direction for research, our method can be used in clustering different cameras and detecting views that are richer in information.

References

[1] B. A. Boghossian and S. A. Velastin. 1999. Motion-based machine vision techniques for the management of large crowds. In Proceedings of the 6th IEEE International Conference on ICECS'99, 2: 961-964. IEEE.

[2] J. Yin, S. Velastin, and A. Davies. 1996. Image processing techniques for crowd density estimation using a reference image. Recent Developments in Computer Vision, pp. 489-498.

[3] A. C. Davies, J. H. Yin, and S. A. Velastin. 1995. Crowd monitoring using image processing. Electronics & Communication Engineering Journal, 7(1): pp. 37-47.
[4] C. Eggert, D. Zecha, S. Brehm, and R. Lienhart. 2017. Improving Small Object Proposals for Company Logo Detection. In Proceedings of the ACM on International Conference on Multimedia Retrieval, pp. 167-174. ACM.
[5] H. Su, H. Yang, and S. Zheng. 2010. The large-scale crowd density estimation based on effective region feature extraction method. In Asian Conference on Computer Vision. Springer.
[6] Y. L. Hou and G. K. Pang. 2011. People counting and human detection in a challenging situation. In IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 41(1): pp. 24-33.
[7] J. Rittscher, P. H. Tu, and N. Krahnstoever. 2005. Simultaneous estimation of segmentation and shape. In IEEE Conference on CVPR, pp. 486-493.
[8] R. Mehran, A. Oyama, and M. Shah. 2009. Abnormal crowd behavior detection using social force model. In IEEE Conference on CVPR, pp. 935-942.
[9] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. 2010. Anomaly detection in crowded scenes. In IEEE Conference on CVPR, pp. 1975-1981.
[10] M. Basseville and I. V. Nikiforov. 1993. Detection of abrupt changes: theory and application. Prentice Hall Englewood Cliffs, Vol. 104.
[11] F. Gustafsson. 1996. The marginalized likelihood ratio test for detecting abrupt changes. In IEEE Transactions on Automatic Control, 41(1): pp. 66-78.
[12] F. Gustafsson and F. Gustafsson. 2000. Adaptive filtering and change detection. New York: Wiley, Vol. 1.
[13] X. Gao, B. Xiao, D. Tao, and X. Li. 2010. A survey of graph edit distance. Pattern Analysis and Applications, 13(1): pp. 113-129.
[14] S. Rajasegarar, C. Leckie, and M. Palaniswami. 2014. Hyperspherical cluster based distributed anomaly detection in wireless sensor networks. Journal of Parallel and Distributed Computing 74.1: 1833-1847.
[15] M. Yang, S. Rajasegarar, A. S. Rao, C. Leckie, and M. Palaniswami. 2016. Anomalous Behavior Detection in Crowded Scenes Using Clustering and Spatio-Temporal Features. In 9th IFIP TC 12 International Conference on IIP. Springer International Publishing, pp. 132-141.
[16] O. Barnich and M. Van Droogenbroeck. 2009. ViBe: a powerful random technique to estimate the background in video sequences. In IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE.
[17] E. V. Cuevas, D. Zaldivar, and R. Rojas. 2005. Kalman Filter for Vision Tracking.
[18] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2): pp. 83-97.
[19] F. Lütteke, X. Zhang, and J. Franke. 2012. Implementation of the hungarian method for object tracking on a camera monitored transportation system. In 7th German Conference on Proceedings of ROBOTIK.
[20] D. Greenhill, J. Renno, J. Orwell and G.A. Jones. 2008. Occlusion analysis: Learning and utilising depth maps in object tracking. Image and Vision Computing 26.3: 430-441.
[21] S. Aminikhanghahi and D. J. Cook. 2017. A survey of methods for time series change point detection. Knowledge and Information Systems, 51(2), 339-367.
[22] E. Zhang and Y. Zhang. 2009. F-measure. In Encyclopedia of Database Systems. Springer US, pp. 1147-1147.