SAIVT-QUT@TRECVid 2012: Interactive Surveillance Event Detection

Jingxin Xu, Simon Denman, Sridha Sridharan, Clinton Fookes
Image and Video Laboratory
Queensland University of Technology
GPO Box 2434, Brisbane 4001, Australia
Emails: {j15.xu, s.denman, s.sridharan, c.fookes}@qut.edu.au

Abstract—

1) **Briefly, what approach or combination of approaches did you test in each of your submitted runs? —- We have only submitted 1 run to detect the events of ‘PeopleMeet’, ‘PeopleSplitUp’ and ‘Embrace’. In our approach, the long video sequence is divided into a set of non-overlapping clips with a small temporal interval. Particle trajectories are approximated directly from the MPEG motion vector. The dense particle trajectories are further segmented into a sparse set of dominant trajectories. A histogram-based feature descriptor is proposed based on the angles among the piecewise trajectory-segments. A modified version of labelled LDA is applied to train a set of topics with a subset of topics reserved for the event of interest, and remainder used for the background activities. In the detection process, the system outputs the likelihood ratio, which can be viewed as a one-dimensional time series signal. The event of interest is detected using a watershed-like algorithm.**
2) **What if any significant differences (in terms of what measures) did you find among the runs? —- We only submitted a single run.**
3) **Based on the results, can you estimate the relative contribution of each component of your system/approach to its effectiveness? —- We have achieved real time performance by directly extracting features from compressed domain and limiting the size of the feature histogram to an acceptable size. We apply a modified version of labelled LDA to learn the feature for the event of interest which helps overcome the problem of the database only being temporally annotated (i.e. no location information). Following detection, the continuous tiny clips are grouped using a watershed-like algorithm to detect the event temporally.**
4) **Overall, what did you learn about runs/approaches and the research question(s) that motivated them? —- Our system achieves a level of performance similar to, or slightly above the state-of-the-art approaches for the detection of the ‘PeopleMeet’ and ‘PeopleSplitUp’ events. However, the results for the detection of the ‘Embrace’ event are poor. This is because the angles between piecewise representative trajectories fail to capture the features for the ‘Embrace’ event well. The proposed approach is also limited by the need to have the persons of interest moving within the sequence, as stationary people do not generate any trajectory features. However, the proposed feature from the compressed domain allows our system to run very fast, such that each iterative search (using pre-computed feature) can be completed within 1 seconds.**

I. Introduction

In this paper, we propose an approach which attempts to solve the problem of surveillance event detection, assuming that we know the definition of the events. To facilitate the discussion, we first define two concepts. The event of interest refers to the event that the user requests the system to detect; and the background activities are any other events in the video corpus. This is an unsolved problem due to many factors as listed below:

1) Occlusions and clustering: The surveillance scenes which are of significant interest at locations such as airports, railway stations, shopping centers are often crowded, where occlusions and clustering of people are frequently encountered. This significantly affects the feature extraction step, and for instance, trajectories generated by object tracking algorithms are usually not robust under such a situation.
2) The requirement for real time detection: The system should process the video fast enough in both of the feature extraction and the detection step to facilitate real time operation.
3) Massive size of the training data set: Suppose there is an event that lasts for 1 minute in a video with a frame rate of 25fps, the number of frames for this events is $60 \times 25 = 1500$. If we want to have a training data set with many positive instances of the event, the video is likely to be very large in size (i.e. hundreds of thousands of frames or more). How to handle such a large data set is a problem frequently encountered in this application.
4) Difficulty in separating the event of interest from background activities: The events of interest often co-exist with a set of background activities. Temporal ground-truth typically very ambiguous, as it does not distinguish the event of interest from a wide range of co-existing background activities. However, it is not practical to annotate the locations of the events in large amounts of video data. This problem becomes more serious in the detection of multi-agent interactions, since the location of these events can often not be constrained to within a bounding box.
5) Challenges in determining the temporal boundaries of the events: An event can occur at any arbitrary time

with an arbitrary duration. The temporal segmentation of events is difficult and ambiguous, and also affected by other factors such as occlusions.

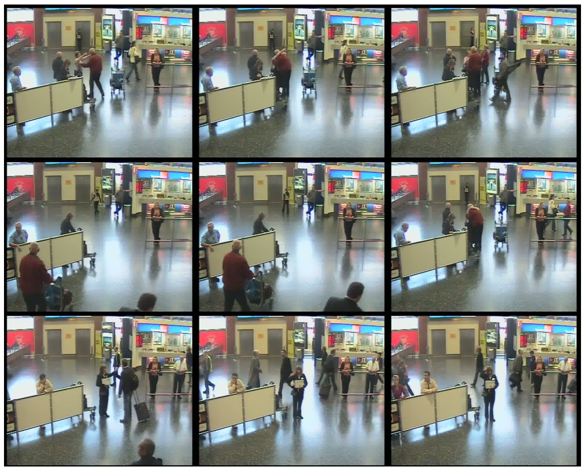


Fig. 1. Sample Images from the Data set (CAM3) for the events of interest. Top: Embrace; Middle: PeopleMeet; Bottom: PeopleSplitUp

Our approach attempts to address all issues above using the TRECVid 2012 SED data set for experiments. Three events are selected to detect, which are Embrace, PeopleMeet and PeopleSplitUp. Figure 1 shows an example of each from the data set. In our system, we use the MPEG motion vector [1] in the compressed domain to replace optical flow as the raw input. The long video sequence is cut into short (i.e. a few seconds) temporal clips. Within a clip, particle trajectories are approximated using MPEG motion vectors. We segment these dense particle trajectories into a sparse set of representative trajectories for the dominant motion flows in the sequence. By extracting the angle between every pair of trajectories, a bag-of-words descriptor is built. We limit the size of vocabulary to ensure computational efficiency.

A modified labelled LDA [2] approach is used to extract the features for the events of interest from a noisy background in the training data set. A watershed-like algorithm is proposed to segment the continuous tiny temporal clips into a large clip and detect the temporal boundaries of the event. This approach addresses all of the problems outlined above:

1) the proposed feature does not rely on object tracking and is robust in crowded scenes with occlusions and clustering of individuals;
2) using MPEG motion vectors allows the data set to be processed rapidly with limited storage and achieve real time performance;
3) the size of the vocabulary is limited, ensuring real time performance and allowing the model to be trained within a short time with limited memory;
4) the smallest unit in our application is a short video clip, which reduces the number of training samples compared to the alternatives of using a frame or a spatio-temporal patch as the smallest unit;
5) the feature for the events of interest are learnt using a modified labelled LDA [2] that is able to handle the events of interest co-existing with the background events, and determine whether the video clip contains the event of interest at a clip level; and
6) the temporal boundaries of events are determined using a watershed-like algorithm to group the continuous small uniform clips that contain the events of interest.

The proposed system can not only be used for real time surveillance event detection, but can also be used as an efficient search engine which allows the user to search for the event of interest and output the times when the query event occurs. For 15 hours video files with preprocessed feature extraction, a result can always be returned within 1 seconds in our experiments. Furthermore, we have achieved promising results when detecting the PeopleMeet and PeopleSplitUp events. It should be noted that although we are participating in the interactive event detection task, our system in fact is more like an application for retrospective event detection. The human operator is only required to tell the computer what event to detect, and the detection process is conducted in a fully automatic manner (i.e. there is no subsequent process where the human operator filters the returned events to remove false alarms).

II. Feature Extraction from the Compressed Domain

This section presents the feature extraction process for our system. In our approach, the MPEG motion vectors are used as the initial data from which to extract the feature [1], [3]. While optical flow can serve the same purpose as the MPEG motion vectors, the MPEG motion vector has several practical advantages.

It is assumed that in a typical real world surveillance system where a CCTV camera network is installed; each camera works simply as a sensor (i.e. data is not processed by the cameras), and data captured by the sensors is transmitted over a network a set of servers for storage and possible further processing (i.e. analytics). The data transmitted over the network is typically encoded in a compressed video format (i.e. MPEG-4). In such a situation, if an analytics algorithm running on the server requires optical flow for feature extraction, there is a large computational cost in extracting the video into an image sequence and computing optical flow. Directly extracting features from the compressed domain saves significant computational cost and storage space.

The first step in the proposed framework is dividing the long entire video sequence into M uniform non-overlapping video clips. The duration of each clip is set much smaller than the usual duration of an event, and following classification a higher level segmentation is performed on these short video clips to group the video clips with the same event of interest.

The TRECVid data set contains video files encoded into MPEG-2 (main profile) format, with the size of GOP the (Group of Pictures) set to 12 frames. A GOP is a fixed length sequence of ordered frames. In the TRECVid, the order is

$IBBPBBPBB$, where the I, P and B indicate I-frames, P-frames and B-frames respectively. Rather than use a single GOP, it is more suitable to set the video clip size to $12 \times N$ frames where N is an integer [1] In our experiments, N is set fo 4, and thus the clip size is 48 frames, less than 2 seconds as the video files have a 25fps frame rate. The motion vector referred to in this paper is defined as the forward motion vectors if the current frame is a P-frame or a B-frame, and the inverse motion vector of the backward motion vector of the preceding B-frame if the current frame is an I-frame.[2] In MPEG-2, the spatial unit of the motion vectors is called a macro-block, which is a 16×16 grid. The pixels in the same macro-block are assumed to be undergoing the same motion. There are also some macro-blocks (intra-blocks) which do not have motion vectors and we assume the velocity is 0 at these locations.

Given a video clip, particles are initialised at the centres of the macro-blocks. Then the particles are propagated along the pixel's motion. This process results in a set of particle trajectories. There are often a lot of noisy trajectories caused by illumination variations and artifacts in the video files, which are removed by thresholding according to the number of stationary points within the trajectory. [3] If more than 25 of the total 48 points are stationary, the trajectory is removed as the motion is considered to be caused by noise.[4] We then remove all stationary points in the remaining trajectories, resulting in the minimum length of a trajectory being $48 - 25 = 23$ points.

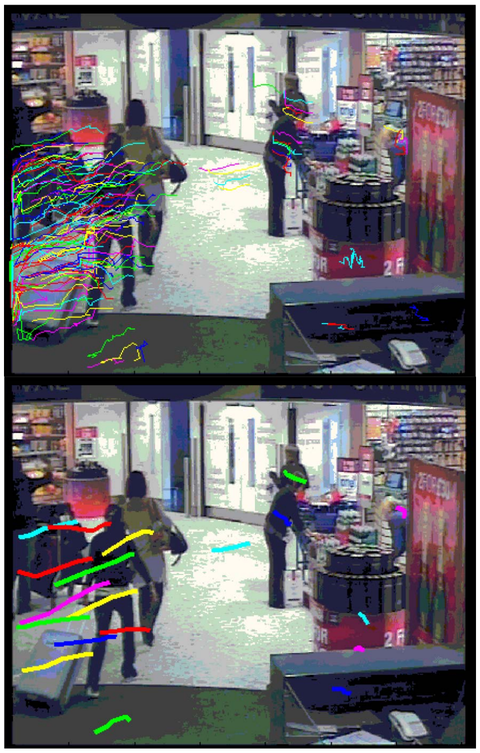


Fig. 2. Visualisation of trajectories generated using MPEG motion vectors. Top: the dense particle trajectories; Bottom: the sparse representative trajectories

The process stated above results in a set of dense trajectories. A clustering algorithm is designed to segment the trajectories into a set of representative trajectories. We first normalize the trajectories to a uniform length, N, and selected key points are sampled uniformly.[5] Then, every trajectory is represented as N key points.

Let a trajectory be represented as $L = \{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$. Now the trajectories are stored in a finite order set, A. Each trajectory is represented as $a_0, a_1, a_2, \ldots, a_N$. The data structure for this set is a linked list. The first trajectory, a_0, is selected and then removed from A. The first representative trajectory, c_0, is set to a_0. We then select the next trajectory from A, which should be a_1. The Euclidean distance between c_0 and a_1, which is denoted as $d(c_0, a_1)$, is computed. If $d(c_0, a_1) < \sigma$, where σ is a threshold; a_1 is said to be active and c_0 is set to $c_0 = c_0 + a_1$. Then, we remove a_1 from A, and select the next trajectory from A. Let a_k be the kth active element from A, we then replace c_0 with $c_0 = (m \times c_0 + a_k)/(m+1)$. This process iterates through all elements from A, and the resultant c_0 is the first representative trajectory. If A is not empty (i.e. $(A \neq \varnothing)$), this procedure is repeated to compute the next representative trajectory. Algorithm 1 describes this process, and Figure 2 shows the visualised results of the dense particle trajectories and the representative trajectories after the clustering algorithm.

Algorithm 1 Trajectory clustering to find a set of representative trajectories

```
$k = 0$
$A$ is a set stored in a linked list;
$A(i)$is the $i$th element from $A$;
$size(A)$ is the size of $A$;
while $(A \neq \varnothing)$
{
    $c_k = A(0)$;
    $m = 1$;
    for $i = 0$ to $size(A)$
    {
     $a_i = A(i)$;
     if $d(c_k, a_i) < \sigma$
          $c_k = (m \times c_k + a_i)/(m-1)$;
    end if
     remove $a_i$ from linked list $A$
    }
    $k = k + 1$;
}
```

[1] Note that 12 is selected as this is the length of the GOP.

[2] Due to the MPEG-2 standard, an I-frame doesn't have motion vectors and the frame before an I frame is guaranteed to be a B-frame.

[3] In a trajectory $L = \{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$, the point (x_k, y_k) is defined as a stationary point if and only if $x_k = x_{k-1}$ and $y_k = yk - 1$, where $k \geq 1$.

[4] The total number of points in a trajectory is 48 indicated by the clip size.

[5] The minimum length of a trajectory is 23, thus the N should not be larger than 23. In our implementation, $N = 5$.

Following trajectory clustering, a bag-of-words descriptor is used to extract features for each video clip. Firstly, each video clip is further divided into K uniform non-overlapping sub-clips (in our implementation $K = 4$), which divides each representative trajectory into K segments. For this reason, $K < N$. As a sub-clip is in a very short duration, the flow segments in a sub-video clip can be viewed as straight lines, and if $K = N - 1$, this will be the case. The angles of every two flow segments in a sub clip are computed, and quantised into 36 bins, together with the distance between the mid-points of every two flow segments. A weighted histogram of angles is built for a sub clip.

Suppose we have two flow segments with an angle of θ between them, and the distance between the mid- points is d. Let h be the weighted histogram we are computing. We first compute the index, i, of θ in h, as $i = \llcorner\theta/(2\pi)\times 36\lrcorner$ (we use the lower bounding integer as the index), and then update the histogram with $h[i] = h[i] + e^{-\frac{d^2}{D}}$, where D is a constant. The histogram will be converted into integer type by using the lower bound. We concatenate the histogram from each sub clip into a single long histogram, preserving the temporal order of the original clip. The histogram is the final extracted feature.

The suitability of this feature descriptor is addressed as below:

1) If two trajectories are too far away in space, their influence is minimal since the weighting decreases with the distance.
2) If two trajectories are moving approximately in a similar direction, their presence will be recorded in the bins close to 0^o.
3) If the angles are much larger than 0^o, it may be indicative that either meeting or splitting is happening.
4) When the 'PeopleMeet' event happens, the distances between the trajectories become smaller and smaller in temporal order, and thus the weight of the observation in the histogram increases; when the 'PeopleSplitUp' event happens, the distances become larger and larger in temporal order, and the weight of the observation decreases. Since we concatenate the histograms in each sub clip into a single histogram in temporal order, the descriptor can distinguish between 'PeopleMeet' and 'PeopleSplitUp'.

Weights used in computing the weighted histogram are determined using a Gaussian smoothing function. After computing the raw Euclidean distance, a normalization process is performed. For an image with size $a \times b$, the maximum length is $\sqrt{a^2 + b^2}$. The raw distance is divided by this factor to normalise distances. We set the variance in the Gaussian function to 0.025 in our experiments.

In order to remove artifacts within the data set, possibly introduced through either network transmission errors or video encoding, we use FFMPEG to re-encode the video and use the re-encoded video files for our experiments. Any remaining artifacts or portions of the videos containing errors are ignored by our algorithm.

III. Learning Features for the Events of Interest using labelled LDA

The temporal annotation of the TRECVid development data set is publicly available. However, the events of interest co-exist together with other activities and the temporal annotation does not distinguish between the events of interest and the background activities (i.e. the annotation does not contain any location information). In this section show we outline how using labelled LDA [2] to learn features for the events of interest can help separate them from the background activities.

Labelled LDA [2] was originally proposed for text processing. This model extends the original Latent Dirichlet Allocation [4] to incorporate labels for each document, and trains a topic for each label. In our application, we simplify the video into a binary case (See Figure 3). Λ is the label, which can be 0 or 1. $\Lambda \equiv 0$ indicates that the video clip does not contain the event of interest and $\Lambda \equiv 1$ indicates that the video clip does contain the event of interest. The K topics are separated into two sets: the B background topics and the $(K - B)$ topics for the events of interest. If the video clip contains the event of interest, all topics are activated; otherwise, only the B background topics are activated.

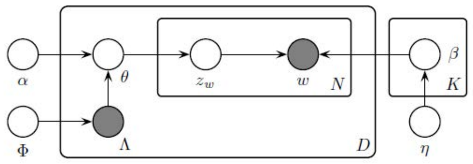


Fig. 3. Labebed LDA [2]

In contrast to the original LDA algorithm, the Markov Chain Monte Carlo (MCMC) algorithm is used for Bayesian inference [5]. In this case, a topic is no longer a multinomial distribution, but a Dirichlet distribution which is driven by the Dirichlet parameter η. This allows us to simply integrate out θ and β and only sample z. Please note that, in our implementation, the hyper parameters α and η are set manually, and β is the parameter we learn. In the MCMC inference, we first perform a random initialisation. Each topic is viewed as a hidden state in a Markov model. A simple regular Markov model is constructed and sequence of transition states is generated from the transition probability matrix in the Markov Model. Given the observations (words), we sample the topics using the approach in [5][2]:

$$P(z_i = j | z_{-i}, w) \propto \frac{n^{w_i}_{-i,j} + \eta}{n_{-i,j} + W\eta} \frac{n^{d_i}_{-i,j} + \alpha}{n^{d_i}_{-i.} + T\alpha}, \quad (1)$$

where n^w_j is the number of times the word, w, has been assigned to the topic, j, and n^d_j is the number of times a word appears in document d. Different from [5][2], T is the number of active topics in a document. For the video clips which do not contain the event of interest, $T = K - B$, otherwise $T = K$.

In the detection process, we use the learned β to compute the likelihood ratio,

$$\frac{P(w,\Lambda=1)}{P(w,\Lambda=0)}=\frac{\prod(\sum P(\Lambda=1)\times P(z|\Lambda=1)\times P(w|z))}{\prod(\sum P(\Lambda=0)\times P(z|\Lambda=0)\times P(w|z))}, \tag{2}$$

where $P(w|z)$ can be obtained directly from β. $P(z|\Lambda)$ is easily computed by marginalising β. $P(\Lambda)$ is computed by counting the number of documents with different labels in the training data set. Then, we can output the likelihood ratio as the score for every video clip. In our experiments, we set the number of topics to 10, with 3 topics for the event of interest.

The usage of labelled LDA for video event detection differs from its' application to text processing [2] in the following ways:

1) In [2], the number of labels can be larger than 2 and each label maps to one topic. In our application, it is a binary classification problem. We only have two labels, and a set of topics. One label maps to all of the K topics, and the other label maps to a subset of these K topics.
2) In [2], it is stated that it was computationally intractable to compute the posterior probability (the probability of a label under the condition of a known bag of words). However, in our application, since we only have two labels, it is computationally tractable to compute the likelihood (joint probability). As a result, we use the likelihood ratio as the score, which is different from the method used in [2].

Recently, [6] proposed a weakly supervised joint topic model to solve the problems of background events and events of interest co-occurring, and some interesting results on traffic surveillance have been demonstrated. This approach differs from our proposed system in that [6] learns a distinctive hyper parameter (the Dirichlet parameter, α) for the event of interest. As a result, both the event of interest and the background activity share the same set of topics, and the event of interest is viewed as a specified distribution of topics. However, a Dirichlet parameter can generate varies Dirichlet distributions, and it only determines the probability of what the Dirichlet distribution (i.e. the topic distribution) looks like. Thus the event of interest represented in this way is more ambiguous compared to our approach. However, [6] addresses the problem of a limited number of positive training samples. As the set of topics is shared, only one parameter will need to be learnt for the event of interest. The selection of which of these two similar approaches to choose should be governed by the number of positive samples in the training data set. Due to the large number of positive examples in the TRECVid data set, the approach in [6] is not suitable for the TRECVid data set.

IV. Temporal Event Detection

The sequence of likelihood ratios is viewed as a one dimensional time series signal. To detect the event of interest and decide the temporal boundaries, we need to detect a signal in this time series data stream. The larger the likelihood ratio is, more likely that the short video clip contains the event of interest.

We first detect the local maxima which are above a threshold, γ. The number of local maxima is the number of events, and these local maxima are the final likelihood scores for the event. To determine the boundary of the events, we set a lower threshold and use a watershed-like algorithm, where the active local maxima are the seeds and we flood the are surrounding the maxima to reach the lower threshold (See Figure 4). In order to reduce the computational time in the detection step, we simplify the detection of boundaries for two merged detected events as follows: the flooding process stop when the boundary score is higher than γ. [6]

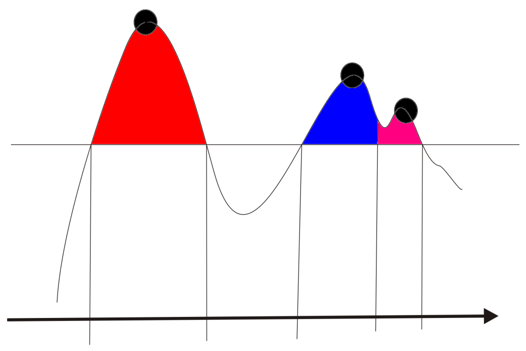


Fig. 4. Temporal Event Detection - Locating the temporal boundaries of the events.

In this way, we can perform a temporal event detection and output the scores and times (start and end points) of the event. It should be noted that, we do not explicitly attempt to detect the beginning and end of an event, and this approach is intended to avoid these situations:

1) the system detects a single event as multiple disjoint events; and
2) the system detects merges multiple events into only a single detected event.

Finally, the detected video segments are sorted by their scores in decreasing order. The system returns a list of ranked video segments (the filename and time spans) and their scores as the output. [7]

V. Experimental Evaluation

All experiments are conducted on a single core of a 2.66Ghz Intel Xeon processor (i.e. we do not use multiple threads). Our proposed approach is implemented using in C++, making use of the VXL and ffmpeg libraries. All video files in the development data set are used, however the video files are separated into 5 subsets for the five different camera views and a model is trained for each camera.

[6]The traditional watershed algorithm finds the local minimum and floods from the bottom to an upper threshold. Our algorithm is an inverse procedure since it is seeded by the local maxima. Meanwhile, we incorporate some simplifications to handle merged segments. This is why we call it watershed-like, as it is similar to and motivated by the watershed algorithm, but with significant differences.

[7]The distinctive large and small values are flattened into the range $[0.08, 2]$, and then normalised into the range $[0, 1]$.

The computational cost of feature extraction depends on the density of the crowd, as crowded scenes will usually result in more particle trajectories and more representative trajectories. To test the speed of the feature extraction, we randomly select 10 video files from the training data set. Each video file lasts for approximately 2 hours. For all of the files, the CPU time required for feature extraction is less than 1 hour, and the maximum time from these 10 video files is 45:04. This demonstrates that our system can perform extract features in real time. Once the features have been extracted, the elapsed time for each iterative search is within a 1 second (from the time that the user tells the system what event to search for, to the time that the system outputs the ranked list of video segments of interest).

We conduct ten search trials for each event to demonstrate the search speed. For the 'PeopleMeet' event, the elapsed time ranges from 0.29s to 0.3s; for the 'PeopleSplitUp' event, this search time required is between 0.3s and 0.32s; and for the 'Embrace' event, this search time ranges from 0.28s to 0.35s. This indicates that our system can be used as either a real time search engine, or a real time, alarm generating surveillance system. Figure 5 shows a demonstration of a query in our software system. [8]

Figure 6 shows the DET curve for our proposed system. Our system performs better for the 'PeopleMeet' and 'PeopleSplitUp' events, than for the 'Embrace' event. This is because our feature descriptor is built by using the angles of every particle trajectory, which is naturally a better fit for the 'PeopleMeet' and 'PeopleSplitUp' events. The performance gap between the 'PeopleMeet' and 'PeopleSplitUp' events can be attributed to the parameters set for the system, in particular the number of topics. The frequency of the 'PeopleMeet' event is 29.46 ipH (instances per hour), while the frequency of 'PeopleSplitUp' is 12.27 ipH. The number of topics for the event of interest should be higher for the 'PeopleMeet' event to reflect the fact that the event is more common, and thus may occur with greater variation. However, for all events the same parameters (i.e. number of topics are used. If the parameters for the detection of 'PeopleSplitUp' is appropriate for the data set, then the parameters for 'PeopleMeet' is are likely to result in under fitting occurring.

The system performance is evaluated by the Normalized Detection Cost Rate (NDCR), where a lower NDCR indicates better performance. The actual NDCR is the NDCR at the operating point used for the submitted the results, and the min NDCR is the NDCR at the selected operation point that maximises the performance, found by searching the DET curve. Compared to the results in the preceding (2011) TRECVid evaluation [7], our system has a lower actual NDCR (0.8898) and a lower min NDCR (0.8482) for the 'PeopleSplitUp' event that the first place time from 2011 (Team TokyoTech-Canon [8] [9], actual NDRC = 0.9099, min NDRC = 0.9066). The performance of the detection of the 'PeopleMeet' event falls within the middle of the field in the 2011 evaluation, where the leading system was proposed by Team PKUNEC [9].

Both of the approaches proposed by [8] and [9] are based around a person detection and tracking framework. Then human's trajectories are used as the feature for a classifier. Compared to our feature, [8] and [9] will have errors when the detection or tracking algorithm fails. Meanwhile, they haven't addressed the issue of filtering out the background activities that co-exist with the event of interest. However, compared to [8] and [9], the feature in our approach will fail in situations where only one of the two persons involved in the activity is moving (a trajectory will not be detected for the stationary person). For instance, it is possible for one person to be standing in the scene waiting for another person to meet him/her. In such a situation, there will be no converging trajectories for this event.

Table I shows the results compared with others from TRECVid SED 2012 [10]. We compare our DCRs for the three events with the average DCRs over all teams. Though we participated in Interactive SED, our system in fact matches the requirements of Retrospective SED. Thus in Table I, both the average DCRs for Interactive SED and Retrospective SED are shown. The performance of our system for 'PeopleSplitUp' is clearly above average, while the performance of 'PeopleMeet' is close to the average performance. The performance for the 'Embrace' event is well below the state of the art.

CONCLUSION

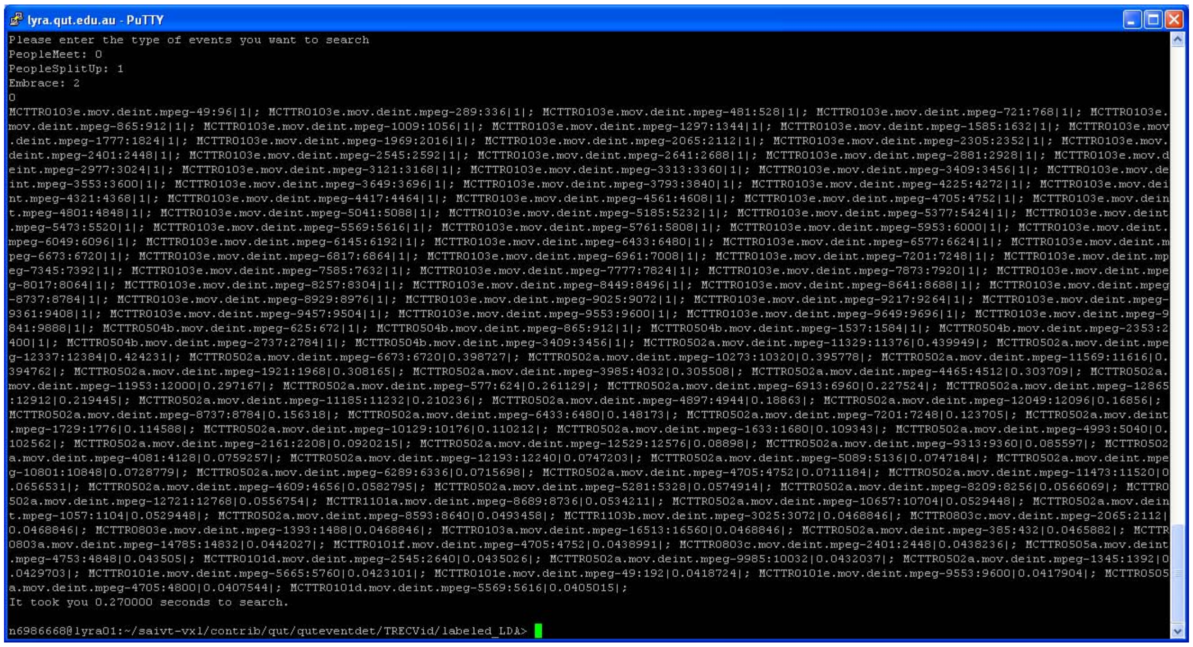
In this paper, we have proposed a novel approach for surveillance event detection. Our approach makes use of the MPEG motion vector as the raw input for feature extraction to ensure real time performance. Our feature does not rely on object tracking, and thus is robust to crowded scenes. We use a modified version of labelled LDA to learn the features for the event of interest, given that we only have the temporal annotation for the training data set. We have also addressed the problem of temporal segmentation and detection for the event of interest through the use of watershed like algorithm. We achieve better or similar results compared to state of the art approaches from the TRECVid SED 2011 for the 'PeopleSplitUp' and 'PeopleMeet' event detection. It is also worth noting that our system is capable of real time performance, either for a retrospective search tasks or for continuous operation from live footage. Despite the promising results however, the accuracy is still far from being useful in a real world application.

ACKNOWLEDGMENT

This research was supported by the Australian Research Council's Linkage Project "Airports of the Future" (LP0990135). The authors also acknowledge the contribution made by the many aviation industry stakeholders also involved in this project. The authors would also like to thank Dr. Patrick Lucey from Disney Research, Pittsburgh, and Dr. Vikas Reddy from Queensland University of Technology for technical discussions.

[8] To facilitate the illustration, the threshold is set lower than the one used in the experiments for our submission so that there are fewer segments detected.

[9] 1st place means the lowest Actual NDCR



```
Please enter the type of events you want to search
PeopleMeet: 0
PeopleSplitUp: 1
Embrace: 2
0
MCTTR0103e.mov.deint.mpeg-49:96|1|; MCTTR0103e.mov.deint.mpeg-289:336|1|; MCTTR0103e.mov.deint.mpeg-481:528|1|; MCTTR0103e.mov.deint.mpeg-721:768|1|; MCTTR0103e.
mov.deint.mpeg-865:912|1|; MCTTR0103e.mov.deint.mpeg-1009:1056|1|; MCTTR0103e.mov.deint.mpeg-1297:1344|1|; MCTTR0103e.mov.deint.mpeg-1585:1632|1|; MCTTR0103e.mov
.deint.mpeg-1777:1824|1|; MCTTR0103e.mov.deint.mpeg-1969:2016|1|; MCTTR0103e.mov.deint.mpeg-2065:2112|1|; MCTTR0103e.mov.deint.mpeg-2305:2352|1|; MCTTR0103e.mov.
deint.mpeg-2401:2448|1|; MCTTR0103e.mov.deint.mpeg-2545:2592|1|; MCTTR0103e.mov.deint.mpeg-2641:2688|1|; MCTTR0103e.mov.deint.mpeg-2881:2928|1|; MCTTR0103e.mov.d
eint.mpeg-2977:3024|1|; MCTTR0103e.mov.deint.mpeg-3121:3168|1|; MCTTR0103e.mov.deint.mpeg-3313:3360|1|; MCTTR0103e.mov.deint.mpeg-3409:3456|1|; MCTTR0103e.mov.de
int.mpeg-3553:3600|1|; MCTTR0103e.mov.deint.mpeg-3649:3696|1|; MCTTR0103e.mov.deint.mpeg-3793:3840|1|; MCTTR0103e.mov.deint.mpeg-4225:4272|1|; MCTTR0103e.mov.dei
nt.mpeg-4321:4368|1|; MCTTR0103e.mov.deint.mpeg-4417:4464|1|; MCTTR0103e.mov.deint.mpeg-4561:4608|1|; MCTTR0103e.mov.deint.mpeg-4705:4752|1|; MCTTR0103e.mov.dein
t.mpeg-4801:4848|1|; MCTTR0103e.mov.deint.mpeg-5041:5088|1|; MCTTR0103e.mov.deint.mpeg-5185:5232|1|; MCTTR0103e.mov.deint.mpeg-5377:5424|1|; MCTTR0103e.mov.deint
.mpeg-5473:5520|1|; MCTTR0103e.mov.deint.mpeg-5569:5616|1|; MCTTR0103e.mov.deint.mpeg-5761:5808|1|; MCTTR0103e.mov.deint.mpeg-5953:6000|1|; MCTTR0103e.mov.deint.
mpeg-6049:6096|1|; MCTTR0103e.mov.deint.mpeg-6145:6192|1|; MCTTR0103e.mov.deint.mpeg-6433:6480|1|; MCTTR0103e.mov.deint.mpeg-6577:6624|1|; MCTTR0103e.mov.deint.m
peg-6673:6720|1|; MCTTR0103e.mov.deint.mpeg-6817:6864|1|; MCTTR0103e.mov.deint.mpeg-6961:7008|1|; MCTTR0103e.mov.deint.mpeg-7201:7248|1|; MCTTR0103e.mov.deint.mp
eg-7345:7392|1|; MCTTR0103e.mov.deint.mpeg-7585:7632|1|; MCTTR0103e.mov.deint.mpeg-7777:7824|1|; MCTTR0103e.mov.deint.mpeg-7873:7920|1|; MCTTR0103e.mov.deint.mpe
g-8017:8064|1|; MCTTR0103e.mov.deint.mpeg-8257:8304|1|; MCTTR0103e.mov.deint.mpeg-8449:8496|1|; MCTTR0103e.mov.deint.mpeg-8641:8688|1|; MCTTR0103e.mov.deint.mpeg
-8737:8784|1|; MCTTR0103e.mov.deint.mpeg-8929:8976|1|; MCTTR0103e.mov.deint.mpeg-9025:9072|1|; MCTTR0103e.mov.deint.mpeg-9217:9264|1|; MCTTR0103e.mov.deint.mpeg-
9361:9408|1|; MCTTR0103e.mov.deint.mpeg-9457:9504|1|; MCTTR0103e.mov.deint.mpeg-9553:9600|1|; MCTTR0103e.mov.deint.mpeg-9649:9696|1|; MCTTR0103e.mov.deint.mpeg-9
841:9888|1|; MCTTR0504b.mov.deint.mpeg-625:672|1|; MCTTR0504b.mov.deint.mpeg-865:912|1|; MCTTR0504b.mov.deint.mpeg-1537:1584|1|; MCTTR0504b.mov.deint.mpeg-2353:2
400|1|; MCTTR0504b.mov.deint.mpeg-2737:2784|1|; MCTTR0504b.mov.deint.mpeg-3409:3456|1|; MCTTR0502a.mov.deint.mpeg-11329:11376|0.439949|; MCTTR0502a.mov.deint.mpe
g-12337:12384|0.424231|; MCTTR0502a.mov.deint.mpeg-6673:6720|0.398727|; MCTTR0502a.mov.deint.mpeg-10273:10320|0.395778|; MCTTR0502a.mov.deint.mpeg-11569:11616|0.
394762|; MCTTR0502a.mov.deint.mpeg-1921:1968|0.308165|; MCTTR0502a.mov.deint.mpeg-3985:4032|0.305508|; MCTTR0502a.mov.deint.mpeg-4465:4512|0.303709|; MCTTR0502a.
mov.deint.mpeg-11953:12000|0.297167|; MCTTR0502a.mov.deint.mpeg-577:624|0.261129|; MCTTR0502a.mov.deint.mpeg-6913:6960|0.227524|; MCTTR0502a.mov.deint.mpeg-12865
:12912|0.219445|; MCTTR0502a.mov.deint.mpeg-11185:11232|0.210236|; MCTTR0502a.mov.deint.mpeg-4897:4944|0.18863|; MCTTR0502a.mov.deint.mpeg-12049:12096|0.16856|;
MCTTR0502a.mov.deint.mpeg-8737:8784|0.156318|; MCTTR0502a.mov.deint.mpeg-6433:6480|0.148173|; MCTTR0502a.mov.deint.mpeg-7201:7248|0.123705|; MCTTR0502a.mov.deint
.mpeg-1729:1776|0.114588|; MCTTR0502a.mov.deint.mpeg-10129:10176|0.110212|; MCTTR0502a.mov.deint.mpeg-1633:1680|0.109343|; MCTTR0502a.mov.deint.mpeg-4993:5040|0.
102562|; MCTTR0502a.mov.deint.mpeg-2161:2208|0.0920215|; MCTTR0502a.mov.deint.mpeg-12529:12576|0.08898|; MCTTR0502a.mov.deint.mpeg-9313:9360|0.085597|; MCTTR0502
a.mov.deint.mpeg-4081:4128|0.0759257|; MCTTR0502a.mov.deint.mpeg-12193:12240|0.0747203|; MCTTR0502a.mov.deint.mpeg-5089:5136|0.0747184|; MCTTR0502a.mov.deint.mpe
g-10801:10848|0.0728779|; MCTTR0502a.mov.deint.mpeg-6289:6336|0.0715698|; MCTTR0502a.mov.deint.mpeg-4705:4752|0.0711184|; MCTTR0502a.mov.deint.mpeg-11473:11520|0
.0656531|; MCTTR0502a.mov.deint.mpeg-4609:4656|0.0582795|; MCTTR0502a.mov.deint.mpeg-5281:5328|0.0574914|; MCTTR0502a.mov.deint.mpeg-8209:8256|0.0566069|; MCTTR0
502a.mov.deint.mpeg-12721:12768|0.0556754|; MCTTR1101a.mov.deint.mpeg-8689:8736|0.0534211|; MCTTR0502a.mov.deint.mpeg-10657:10704|0.0529448|; MCTTR0502a.mov.dein
t.mpeg-1057:1104|0.0529448|; MCTTR0502a.mov.deint.mpeg-8593:8640|0.0493458|; MCTTR1103b.mov.deint.mpeg-3025:3072|0.0468846|; MCTTR0803c.mov.deint.mpeg-2065:2112|
0.0468846|; MCTTR0803e.mov.deint.mpeg-1393:1488|0.0468846|; MCTTR0103a.mov.deint.mpeg-16513:16560|0.0468846|; MCTTR0502a.mov.deint.mpeg-385:432|0.0465882|; MCTTR
0803a.mov.deint.mpeg-14785:14832|0.0442027|; MCTTR0101f.mov.deint.mpeg-4705:4752|0.0438991|; MCTTR0803c.mov.deint.mpeg-2401:2448|0.0438236|; MCTTR0505a.mov.deint
.mpeg-4753:4848|0.043505|; MCTTR0101d.mov.deint.mpeg-2545:2640|0.0435026|; MCTTR0502a.mov.deint.mpeg-9985:10032|0.0432037|; MCTTR0502a.mov.deint.mpeg-1345:1392|0
.0429703|; MCTTR0101e.mov.deint.mpeg-5665:5760|0.0423101|; MCTTR0101e.mov.deint.mpeg-49:192|0.0418724|; MCTTR0101e.mov.deint.mpeg-9553:9600|0.0417904|; MCTTR0505
a.mov.deint.mpeg-4705:4800|0.0407544|; MCTTR0101d.mov.deint.mpeg-5569:5616|0.0405015|;
It took you 0.270000 seconds to search.

n6986668@lyra01:~/saivt-vxl/contrib/qut/quteventdet/TRECVid/labeled_LDA>
```

Fig. 5. A demonstration of an experiment run. The system first asks the user to select the event to detect. In this demonstration, the user selects ‘PeopleMeet’. Then the system automatically outputs the segments of interest in order of relevance, with a format for $filename - start_frame : end_frame|score|$. Finally the search time is shown. In this example, the search time is 0.27 seconds.

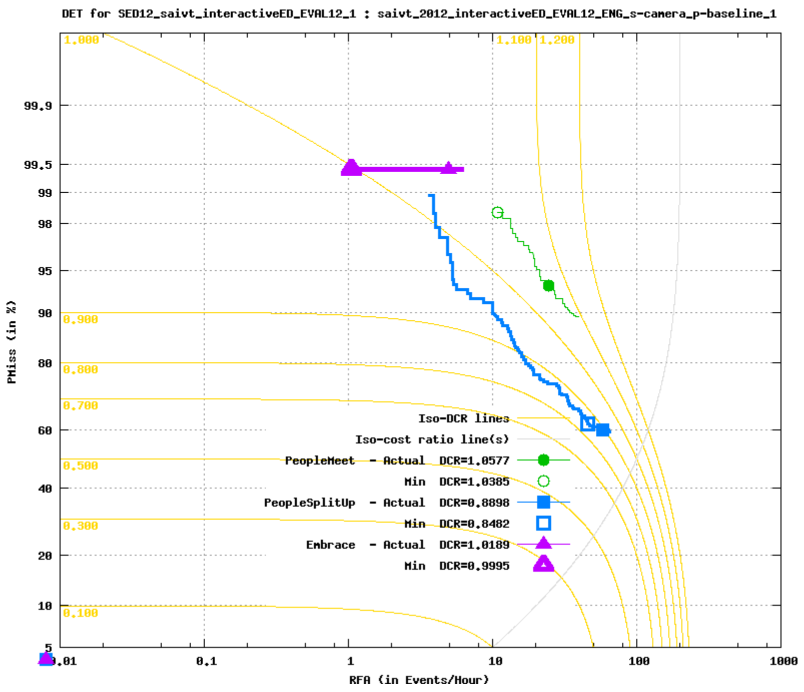


Fig. 6. DET curve for our proposed system

	PeopleMeet	PeopleSplitUp	Embrace
Average DCRs of repSED	1.0834	0.9953	0.9562
Average DCRs of interactiveSED	1.0358	0.9183	0.8640
DCRs of Our Approach	1.0577	0.8898	1.0189

TABLE I
DCRS FOR OUR PROPOSED APPROACH COMPARED TO OTHER PARTICIPANTS OF TRECVID SED 2012.

REFERENCES

[1] P. Tudor, "Mpeg-2 video compression," *Electronics Communication Engineering Journal*, vol. 7, no. 6, pp. 257 – 264, dec 1995.

[2] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora," pp. 248–256, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699510.1699543

[3] C.-W. Su, H.-Y. Liao, H.-R. Tyan, C.-W. Lin, D.-Y. Chen, and K.-C. Fan, "Motion flow-based video retrieval," *Multimedia, IEEE Transactions on*, vol. 9, no. 6, pp. 1193 –1201, oct. 2007.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, March 2003. [Online]. Available: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993

[5] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *PNAS*, vol. 101, no. suppl. 1, pp. 5228–5235, 2004.

[6] T. Hospedales, J. Li, S. Gong, and T. Xiang, "Identifying rare and subtle behaviors: A weakly supervised joint topic model," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 12, pp. 2451 –2464, dec. 2011.

[7] P. Over, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, and G. Quéenot, "Trecvid 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," 2011.

[8] Y. K. K. S. N. Inoue, T. Wada, "Tokyotech+canon at trecvid 2011," 2011.

[9] T. X. Z. X. P. P. Y. W. Y. T. H. Z. F. W. S. T. G. L. W. Z. X. Fang, C. Su, "Pku-nec @trecvid2011 sed: Sequence-based event detection in surveillance video," 2011.

[10] P. Over, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, and G. Quéenot, "Trecvid 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," 2012.