CCTV Video Summarisation

AS Karthik*, Abhishek Krishna†, Gagan Deep G‡ and Dr. Ramakanth Kumar P§
Department of Computer Science, RVCE
Bangalore
Email: {*askarthik.cs15, †abhishekkrishna.cs15, ‡gagandeepg.cs15, ‡ramakanthkumarp, }@rvce.edu.in

***Abstract*—The use of CCTVs has been on the steady rise in in the last decade, being used for the security of public and private establishments alike. With increasing crime rates in cities, there is a need for a smarter method to survey the video being recorded, rather than manually going through hours of footage, saving both time and labour. We propose an efficient method to generate a summary of the footage. The project also aims to simplify the task further, by generating video summaries based on user query, thus showing exactly what is required.**

***Index Terms*—CCTV, video summary, computer vision**

I. Introduction

The number of CCTVs surveillance cameras is increasing everyday leading to a huge amount digital video information being captured and stored. Millions of CCTV cameras run 24 hours a day, sometimes even streaming the content over the internet for people to monitor. This data is however in a raw, unprocessed format. In most cases, video with little to no motion is being captured, which wastes lot of storage. The process of watching or analysing the footage is also time consuming and laborious.

This project proposes to build a CCTV video summariser to eliminate this redundancy in stored data and give the user a short summary [5], [4], [3], [2] consisting of important information that is relevant for the specific use case. The system proposes to incorporate a query based summary generation to generate more relevant summaries. Users can choose the required objects and events to generate a short yet precise summary with only relevant information, saving more time.

The process takes place over two phases, the real-time and query phase. The real-time phase reads the CCTV footage, identifies clips of interest, extracts the "flow-tubes", and stores them after tagging them with the respective object tags. A query phase would then involve the user selecting the required tags and objects of interest, which are chosen. The tubes are rearranged using simulated annealing algorithm, before being blended into a generated time-lapsed background using an optimal technique like poisson blending.

This method reduces the manual work of going through hours of footage looking for relevant events by automatically creating a summary. It can be mainly used for security purposes, by the police forces for detection of crimes and suspicious activities.

II. Literature Review

In [11], Yael Pritch and Alex Rav-Acha propose a method to effectively generate a synopsis of an endless video stream that can also be used as an index into the main video. An online phase includes tube detection in spatio-temporal domain, insertion of these tubes into an object queue, and removal upon reaching a space limit. The response phase then constructs a time-lapse video of the changing background, selection and stitching of tubes into a coherent video. Min-cut algorithm along with background subtraction has been used for extracting moving objects. Activity, collision and temporal consistency costs have been used a parameters for optimal tube arrangement.

In [10], Shmuel Peleg and Yael Pritch, have presented a dynamic video synopsis technique where most of the activity in the video is condensed by simultaneously showing several actions, even when they originally occurred at different times.

In [13], (CRAM: Compact Representation of Action in Movies), Mikel Rodriguez generates a compact video representation of a long sequence, which while preserving the general dynamics of the video features only the essential components. From the given input video, optical flows are generated. These are then represented as vectors in Clifford Fourier domain. Dynamic regions of flow are then identified within the phase spectrum volume. The likelihood of activities of relevance are then computed by correlating it with spatio-temporal maximum average correlation height filters. The final summary is then generated by a temporal-shift optimization. Although this method could detect specific actions, it couldn't keep all the events in the final summary.

Sarit Ratovitch, Avishai Hendel and Shmuel Peleg, in their paper titled Clustered Synopsis of Surveillance Video [8], present a different approach to generating video summaries, based on clustering of similar activities. Objects with similar activities are easy to watch simultaneously, which also makes spotting of outliers easier. This method is also suitable for creation of ground truth data. This paper covered three main topics, the definition of distance between activities, clustering of similar activities and efficient presentation of video summaries using obtained clusters.

In [7], (A Shortest Path Representation for Video Summarization), a new approach for video summarization is presented to select multiple key frames within an isolated video shot where there is camera motion, causing significant scene change. This can be done by determining dominant motion between frame pairs whose similarities are represented using a directed graph. A* algorithm is used to detect the shortest path and designate key frames. The overall set of key frames depict the essential video content and camera motions.

[17] presents a very successful and highly used method for adaptive pixel-level background subtraction. Each pixel has probability density function separately. A pixel is considered to be part of the background if its new value if well described by its density function. This paper was an improvement on previous models which used Gaussian mixture models with efficient update equations.

In [12], an extremely fast object detection model, the YOLO model is described. While prior object detectors used classifiers to detect, this paper proposes object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network is used to predict both bounding boxes and its class probability, making end-to-end optimization easy. Although YOLO makes more localization errors, it is less likely to predict false positives compared to other object detectors like SSD, RCNN and Faster RCNN.

III. Methodology

After going through many papers, the framework described here has been arrived at . It mainly has two phases: Real-Time Phase and Query Phase.

A. Real-time phase

The real-time phase reads the CCTV footage, identifies clips of interest and performs certain image processing algorithms on the footage of interest to extract "flow-tubes" and tags from clips and stores them in a database. [9] The phase is split into the following steps:

1) **Motion Detection**
   Detect motion in the footage and identify any clips with significant motion while disregarding artifacts due to changing environmental conditions and other insignificant disturbances. MOG is used in this step to see if there is any movement, and if there is significant foreground present in an image, decided by a static threshold, the clip is considered to have motion in it. MOG is specifically useful here due to its dynamic nature and adaptability to gradual changes in the environment very quickly and, additionally, availability of efficient implementations of this very effective algorithm.
2) **Background Masking**
   A foreground extractor like a Mixture of Gaussians [17], [1], [14] is used to extract the subjects of interests in the clips identified by motion detection. The same technique used in previous step is also employed here to generate foreground masks and thereby just extracting the foreground. Several techniques were experimented on MOG is the most cost effective and accurate technique available for this use case.
   In Gaussian Mixture Model (GMM) defined as 1, every pixel in the frame is modelled into a Gaussian distribution. Probability of every pixel being the foreground or background is calculated as:

$$P\left(X_t\right) = \sum_{i=1}^{K} \omega_{i,t}.\eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{1}$$

   - X_t: current pixel in frame t
   - K: the number of distributions in the mixture
   - $\omega_{i,t}$: the weight of the k^{th} distribution in frame t
   - $\mu_{i,t}$: the mean of the k^{th} distribution in frame t
   - $\Sigma_{i,t}$: the standard deviation of the k^{th} distribution in frame t

   Where $\eta(X_t, \mu_{i,t}, \Sigma_{i,t})$ is a probability density function defined in 2 as:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\Sigma^{1/2}} exp^{\frac{-1}{2}(X_t-\mu)\Sigma^{-1}(X_t-\mu)} \tag{2}$$

3) **Computation of Objects flow-tubes**
   Flow-tubes are computed from the extracted foreground in previous phase. Flow tubes are extracted by performing morphological operations and several redundant foreground blobs are removed in this step. Furthermore, individual subjects present in each frame are identified, and related back with the subjects present in the previous frame, thereby producing flow-tube arrays.
4) **Object Tagging**
   After actual subjects are identified in the previous phase, the subjects are classified into several popular categories using a popular deep-learning model called "You Only Look Once" model, and these tags are computed.
   A pre-trained 26-layered YOLOv3 model is used as the most common categories present in a common CCTV video footage are already present in the set of categories identifiable on a YOLOv3 trained on the standard COCO dataset.
5) **Metadata storage**
   In this stage, a connection is established to the database and the tags and flow-tubes computed are stored into the database.

B. Query Phase

The query phase processes the user input query, extracts the relevant tubes and generates a relevant summary. This phase is split into the following steps:

- **Tube Selection**
  The user query containing various parameters such as time period, tags and length of summary required are taken from user and relevant flow-tubes are selected from the database.
  This stage is easily implemented by writing logic to create a query with all the parameters the user specifies in the input query.
- **Rearrangement** An optimisation algorithm, in this case simulated annealing [16], is used to rearrange the flow-tubes in the time dimension to produce a summary of the desired length.
  While there are several heuristic based search algorithms are recommended for these purposes by different authors, simulated annealing remains to be the most successful and most popularly cited method. Hence, simulated annealing with a custom cost function has been implemented based on the needs.

C. Cost Function

The heart of the project resides in the rearrangement phase. A heuristic based approach has been adopted to solve this NP combinatorial problem. In order to solve a combinatorial problem using an algorithm like simulated annealing [16], an Energy or a Cost function has to be defined, which embodies the various parameters to be optimised. In this case, the two main parameters to optimise are:

- Collision: The amount of collision between events in the rearranged set of events.
- Length: The total length of the generated summary must be as small as possible.

Collision Cost is calculated as:

$$Cost_{Collision} = \frac{Collision}{TotalPixels - Collision} \tag{3}$$

where,

$$Collision = \sum_{i=0}^{N}\sum_{j=i}^{N}\left(\sum_{k=\max(s_i,s_j)}^{\min(e_i,e_j)} T_i[k].T_j[k]\right) \tag{4}$$

$$TotalPixels = \sum_{i=0}^{N}\sum_{j=i}^{N} \left(\sum_{k=\max(s_i,s_j)}^{\min(end_i,end_j)} \sum\Big(T_i[k == w] + \sum T_j[k == w]\Big)\right) \tag{5}$$

e_i: the time at which the clip *i* ends
s_i: the time at which the clip *i* ends
T_i: the 3D array (tube) representing the event in a boolean map format
w: the value of all foreground pixels in the T_i

Length Cost is calculated as:

$$Cost_{length} = \frac{Length - Lowerlimit}{Upperlimit - Lowerlimit} \tag{6}$$

where

$$Length = \max_{\forall i \in \tau}(end_i) - \min_{\forall i \in \tau}(start_i) \tag{7}$$

$$Lowerlimit = \max_{\forall i \in \tau} len_i \tag{8}$$

$$Upperlimit = \sum_{i}^{N} end_i \tag{9}$$

end_i: the time at which the clip *i* ends
$start_i$: the time at which the clip *i* ends

The total cost is given as:

$$TotalCost(W, Cost) = sigmoid(W.Cost^T) \tag{10}$$

where,

- W is weight vector of the form $[Weight_{Collision}, Weight_{Length}]$ assigning different priorities for the two factors
- $Cost$ is Cost vector of the form $[Cost_{Collision}, Cost_{Length}]$
- $sigmoid$ is defined as -

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

- **Time-lapsed background generation** In this step a background is generated based on the time period and summary length required by user.

  A weighted approach, with the periods where there is most activity, is considered more heavily in generating the time-lapsed background.
- **Blending** Poisson blending [15], [6] is used to blend the rearranged flow-tubes with the time-lapsed background to generate the summary video. This summary is then saved onto the user's computer.

IV. Implementation

A. Simulated Annealing Algorithm

Algorithm 1 shows the simulated annealing algorithm which is used for the tube optimisation process.

Result: Optimised Configuration
Let $Config_{current} = Config_{inital}$; **for** $T \leftarrow T_{max}$ **to** T_{min} **do**
  $E_{current} = E(C_{current})$ $C_{next} = next(C_{current})$
  $E_{next} = E(C_{next})$ $\Delta E = E_{next} - E_{curr}$ **if** $\Delta E > \theta$ **then**
    $C_{current} \leftarrow C_{next}$
  else if $e^{\frac{-\Delta E}{T}} > rand(0,1)$ **then**
    $C_{current} \leftarrow C_{next}$
end

Algorithm 1: Simulated Annealing

B. Tube Extraction Algorithm

Algorithm 2 shows the tube extraction algorithm which is used for the extracting the event tubes from the labelled motion volume.

V. Results

Sample video clips in different urban areas with sparse motion and activity were collected, with only 1-2 events occurring once every few seconds. These clips were used for developing and testing the algorithm.

Figure 1 shows some frames from one of our experimental videos. Only 1-2 events are seen occurring at any point of time. People and bikes are the objects that are seen.

Figure 2 shows an output frame generated by our video summariser. Many people and bikes are seen simultaneously,

Result: 3D space-time volume with connected labelled events

```
/* a 3d array representing the
   time-space volume with foreground
   represented as 1 and background as
   0                                  */
```

input: VideoMask

volume ← *empty_list*() *tube_num* ← 0
prev_frame ← *zeros_like*(*VideoMask*[1])
prev_count ← 0
for *FrameinVideoMask* **do**
  labelled_frame, *count* = *findConnectedComponents*(*Frame*)
  for *iinrange*(0, *count*) **do**
    matches ← *empty_list*() **for** *jinrange*(0, *prev_count*) **do**
      if $\frac{Sum((labelled\_frame[k==i]*prev\_frame))}{Sum(prev\_frame)} >$ *Threshold* **then**
        matches.append(*j*)
      end
    end
    matches_dict[*i*] ← *matches*
  end
  prev_frame ← *Frame prev_count* ← *count*
  for *regioninmatch_dict* **do**
    if *len*(*match_dict*[*region*])>1 **then**
      for *indexinmatch_dict*[*region*] **do**
        for *frameinvolume* **do**

```
/* Update labels of tubes
   in previous frame    */
```

          frame[*frame* == *index*] = *tube_num*
        end

```
/* Update the dict      */
```

        match_dict[*region*] = [*tube_num*]
      end
    end
  end

```
/* There are no intersection
   issues; Proceed normally     */
```

  for *regioninmatch_dict* **do**
    cur_img ← *labelled_frame*
    cur_img_output ← *labelled_frame.clone*()
    cur_labels = *empty_list*() **if** *len*(*match_dict*[*region*]) == 0 **then**

```
/* New region found since
   there is no match to any
   previous region          */
```

      tube_num+ = 1
      cur_img_output[*cur_img* == *region*] = *tube_num*
    end
    else

```
/* Some previous region was
   found, relabel the region
   */
```

      cur_img_output[*cur_img* == *region*] = *match_dict*[*region*][0]
    end
  end
  volume.append(*cur_img_output*)
end

Algorithm 2: Tube Extraction

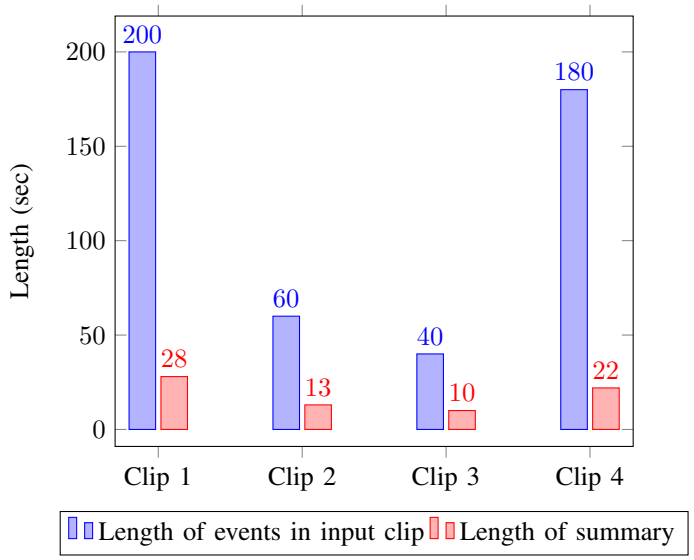
Fig. 1. Selection of input frames from one of our experimental videos

and the timestamp shows the time at which the event occurred in the original input video. It is apparent from this picture that the density of events is dramatically improved.

Fig. 2. A frame from the generated summary video

A. Objective Evaluation Parameters

- **Exhaustiveness of summary** The summariser must retain all the events that occurred or the events of the specified type in the tag-based summary generation.
  Result: All significant events are selected in the events
- **Compression factor** Given by Length of original input/Length of summary. The compression factor must be as high as possible while keeping the overlap factor to a minimum.
  Result: There is a compression factor of 4-7x for our sample videos



- **Overlap factor** A number between 0 and 1 which indicates the amount of overlapping (intersection) between the tubes in the summary video. 0 indicates no overlap and 1 indicates complete overlap of every clip.
  Result: All videos have overlap lower than 0.1

B. Subjective Evaluation Parameters

- **Semantic structure of events** Interacting events (that occur in the same time and space) must be shown together.
  Result: All the interacting events are always shown together
- **Realistic appearance** The events must be extracted and be blended into the background seamlessly and appear realistic.
  Result: Realistic appearance in most cases, with some halo around object when at the edges of the frame, or when there is intersection.

VI. Conclusion

This project aimed to use a novel method to generate video summaries to reduce the amount of time spent in analyzing CCTV video footage. Our implementation of summarization by temporal rearrangement of events improves on other methods of just detecting frames of motion. The tag-based summary selects only the required type of event, reducing the summary length further.

A proof-of-concept has been presented, and it can be extended to be used in commercial CCTV systems with further development.

References

[1] Philippe Loic Marie Bouttefroy, Abdesselam Bouzerdoum, Son Lam Phung, and Azeddine Beghdadi. On the analysis of background subtraction techniques using gaussian mixture models. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4042–4045. IEEE, 2010.

[2] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. Video co-summarization: Video summarization by visual co-occurrence. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[3] Ying Li, Tong Zhang, and Daniel Tretter. An overview of video abstraction techniques. Technical report, Technical Report HPL-2001-191, HP Laboratory, 2001.

[4] Jeho Nam and Ahmed H Tewfik. Video abstract of video. In *1999 IEEE Third Workshop on Multimedia Signal Processing (Cat. No. 99TH8451)*, pages 117–122. IEEE, 1999.

[5] Jung Hwan Oh, Quan Wen, Sae Hwang, and Jeongkyu Lee. Video abstraction. In *Video data management and information retrieval*, pages 321–346. IGI Global, 2005.

[6] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 313–318, 2003.

[7] Sarah V Porter, Majid Mirmehdi, and Barry T Thomas. A shortest path representation for video summarisation. In *12th International Conference on Image Analysis and Processing, 2003. Proceedings.*, pages 460–465. IEEE, 2003.

[8] Yael Pritch, Sarit Ratovitch, Avishai Hendel, and Shmuel Peleg. Clustered synopsis of surveillance video. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 195–200. IEEE, 2009.

[9] Yael Pritch, Alex Rav-Acha, Avital Gutman, and Shmuel Peleg. Webcam synopsis: Peeking around the world. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[10] Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. Nonchronological video synopsis and indexing. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1971–1984, 2008.

[11] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 435–441. IEEE, 2006.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[13] Mikel Rodriguez. Cram: Compact representation of actions in movies. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3328–3335. IEEE, 2010.

[14] Jian Sun, Weiwei Zhang, Xiaoou Tang, and Heung-Yeung Shum. Background cut. In *European Conference on Computer Vision*, pages 628–641. Springer, 2006.

[15] Richard Szeliski, Matthew Uyttendaele, and Drew Steedly. Fast poisson blending using multi-splines. In *2011 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2011.

[16] Xin Yao. A new simulated annealing algorithm. *International Journal of Computer Mathematics*, 56(3-4):161–168, 1995.

[17] Zoran Zivkovic et al. Improved adaptive gaussian mixture model for background subtraction. In *ICPR (2)*, pages 28–31. Citeseer, 2004.