

A NOVEL VIDEO MINING SYSTEM

Arasanathan Anjulan and Nishan Canagarajah

Department of Electrical and Electronic Engineering, University of Bristol, UK

ABSTRACT

This paper describes a novel object mining system for videos. An algorithm published in a previous paper by the authors is used to segment the video into shots and extract stable tracks from them. A grouping technique is introduced to combine these stable tracks into meaningful object clusters. These clusters are used in mining similar objects. Compared to other object mining systems, our approach mines more instances of similar objects in different shots. The proposed framework is applied to a full length feature film and improved results are shown.

Index Terms— object mining, feature extraction, object clustering

1. INTRODUCTION

Data mining is the process of analyzing data from different perspectives and to find useful patterns or information which can be used in a variety of applications. Text data mining is popular among researchers and have a large application domain varying from business databases to web applications [1,2]. However despite the vast amount of visual information present in digital media, the research done in visual data mining is limited. There are several reasons for this. The semantic gap between the low level visual features (such as colour, texture and motion) and a higher level user domain (such as object and event identification) is one of the main curdles for the development of a visual data mining system. Moreover, there is no meaningful clustering or segmentation method that can be universally applied to all kind of visual media. In text, words can be easily clustered into groups. However this is difficult for images and videos. In visual media an object may appear in different imaging conditions (different camera angles, zoom positions, or lighting conditions) and can also be occluded. All these variations make visual data mining more challenging compared to text data mining.

Existing work in video data mining can be divided into two categories: mining similar motion patterns and mining similar objects. The first type of systems use motion information to mine similar event patterns or identify peculiar events.

Arasanathan Anjulan and Nishan Canagarajah are with the Department of Electrical and Electronic Engineering, University of Bristol, Bristol, BS8 1UB, UK (e-mail: A.Anjulan@bristol.ac.uk, Nishan.Canagarajah@bristol.ac.uk).

The second category systems aim to group frequently appearing objects in videos. We define different appearances of the same object in different parts of the video as different instances of that object. The purpose of the object mining system is to group these different instances of the same object. The problem is difficult because the object can appear in different ways in different parts of the video due to different imaging conditions, lightening conditions, back grounds and occlusions.

In [3,4], authors use vector quantize and spatial neighborhood techniques to mine frequently appearing objects in video. Their system is based on key frames and any interesting objects appearing outside the key frame will be missed and any failure in key frame extraction will affect the mining system. Moreover different spatial neighborhood sizes needed to mine different objects.

This paper use an earlier work published [5] by the authors to segment the video into shots¹ and extract features from stable local invariant tracks. These features are grouped into clusters to represent relevant objects. Each cluster approximately corresponds to an object in a shot. These object clusters may contain a number of similar instances of the same object, and these instances are grouped together in the mining stage. The effectiveness of the system is demonstrated by performing object mining in a full length feature film. The rest of this paper is organized as follows. The proposed mining frame work is described in section 2 and results from a real movie are presented in section 3. We conclude in section 4 with suggestions for future work.

2. PROPOSED FRAMEWORK

2.1. Shot Segmentation and Representation

The first stage of the system divide the video into a number of meaningful shots, and obtain representation for objects that are present in the shots. Both of these tasks are performed simultaneously using the local invariant features obtained at each frame. An algorithm previously published by the authors [5] are used for this purpose. In this paper, MSER [6] is used to extract local invariant regions (LIRs) and an ellipse fitting algorithm is used to approximately fit an ellipse to each of

¹A shot is the basic element in a video and can be defined as an uninterrupted sequence of frames recorded from a single camera operation.

these regions. SIFT [7] is used to assign a 128 dimensional descriptor for each of the extracted ellipse regions. Although we propose to use MSER and SIFT, the system can work with any combination of region extractor and descriptor.

Algorithm 1 Obtaining object clusters from tracks in a shot

Notation \mathbf{m}_c^f center of object cluster c at frame f
 Ψ_c^f covariance of object cluster c at frame f
 \mathbf{m}_t^f center of track t at frame f

1. Initialize

The algorithm is initialized with a single cluster with the longest track in the shot.

2. Iterate

a Assignment of Tracks

For each track t in the shot, we calculate the distances to each of the current object cluster \mathcal{O}_c

$$D_{ct} = \sum_f D_{ct}^f = \sum_f \left(\mathbf{m}_c^f - \mathbf{m}_t^f \right)^T \left(\Psi_c^f \right)^{-1} \left(\mathbf{m}_c^f - \mathbf{m}_t^f \right) \quad (1)$$

and the closest object cluster c^* is given by

$$c^* = \underset{c=1}{\operatorname{argmin}}^K D_{ct} \quad (2)$$

if $D_{c^*t} < \tau_r$, then the track t is assigned to the object cluster c^* , otherwise a new object cluster is started with track t as it's first member.

b Estimating the parameters of object clusters

for each cluster object cluster \mathcal{O}_c and for each frame associated with that cluster, calculate the following parameters

$$\mathbf{m}_c^f = \frac{1}{N_{\mathcal{O}}} \sum_{t \in \mathcal{O}_c} \mathbf{m}_t^f \quad (3)$$

$$\Psi_c^f = \frac{1}{N_{\mathcal{O}}} \sum_{t \in \mathcal{O}_c} \left(\mathbf{m}_c^f - \mathbf{m}_t^f \right)^T \left(\mathbf{m}_c^f - \mathbf{m}_t^f \right) \quad (4)$$

3. Convergence

The algorithm is terminated when there is no change in assignment of tracks.

We briefly explain the algorithm proposed in [5]. The shot boundaries are detected based on the number of LIRs from the current frame that are matched with the LIRs from the next frame. A continuous matching of these LIRs across a number of frames is considered as a *track*. A track can start, either from a local region detected at the first frame of the shot or from a local invariant regions at intermediate frames which

are not matched to any of the LIRs in the previous frame. Once a shot boundary is detected we obtain a meaningful representation of the shot by obtaining a collective descriptor for each of the track by averaging the descriptors of the local invariant regions which are associated with that track. The discontinuous nature (due to occlusion or failure of the MSER algorithm due to extreme conditions) of the tracks are handled by further comparing each track to all other mutually exclusive (in terms of frames) tracks within the shot (using the L2 distance between the average descriptors of the tracks). If that distance is less than a threshold, then the two tracks are joined together and a new average descriptor is calculated. A shot is now represented by the average shift descriptors of the detected LIR tracks within that shot. Once the track descriptors are calculated for all the shots of the video, each track is compared with the rest of the tracks, and any track that matches more than a threshold number of tracks, is removed. This will help to remove very common tracks which frequently appear in all the shots. This step will enable to reduce the mismatches among the shots caused by these common tracks. This is similar to the stop list exploited by [3].

2.2. Grouping of Tracks

In the previous section we briefly described how local invariant regions are connected from frame to frame to form tracks within a shot. While track descriptors are sufficient for some applications such as scene retrieval [5], further refinement is needed for object mining and other applications. Hence the next important step is the grouping of tracks from a shot into clusters that approximately correspond to a number of objects in that shot. We define two tracks belong to the same object in a shot if the spatial distance between the tracks are small and approximately constant in the frames where they both appear (common frames to both tracks). The task of grouping the tracks into meaningful clusters is difficult due to the discontinuous nature of the tracks. Moreover any clustering algorithm should be able to handle the size variation of an object in different shots. It is not straight forward to use standard clustering algorithms to achieve these goals. This section describes a heuristic procedure which works in practice.

We start by defining a distance measure between an object cluster and a track in a shot (equation 1, Algorithm box 1). This distance is essentially an average of distances between the cluster center and track center in the shared frames, normalized by the covariance matrices of the cluster at each frame. Given a number of object clusters, a track is assigned to the cluster with a minimum distance, if that minimum distance is below a threshold value. Otherwise a new cluster is started with the track as it's first member. Cluster centers and covariance are recalculated after we have completed the assignment of all the tracks. The procedure is repeated until convergence. It is important to normalize the cluster-to-track distance by the covariance matrix of the cluster to handle

the scale variation of objects in different shots. For example, without this normalization a face appearing in a long shot will be clustered into a single object, whereas the same face in a close shot may be grouped into a number of different object clusters with eyes and nose appearing in separate clusters.

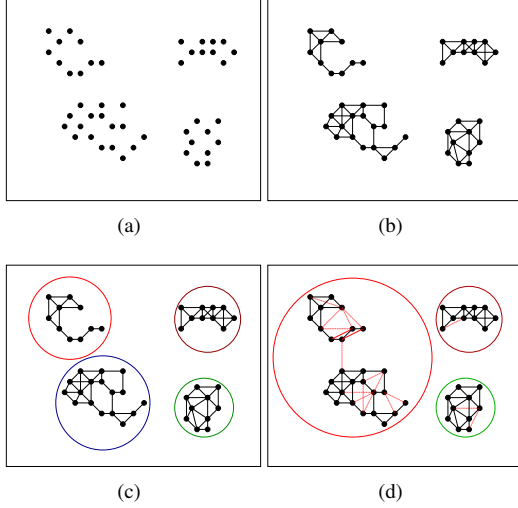


Fig. 1. Grouping of object clusters. (a) Each object cluster is considered as a vertex in a graph. (b) Two vertices are connected if the matching ratio of the corresponding object clusters is above a threshold τ_{ob} (see text). (c) Mutually exclusive groups of connected vertices are identified and classified as object groups using standard graph connected components algorithm. (d) Additional connections, introduced due to a smaller τ_{ob} , are displayed in red color. This leads to a merging of two, previously separate, object groups into a single object group. A smaller τ_{ob} will give fewer and larger object groups. This is useful in object classification applications.

2.3. Object Mining

The previous sub-section described a scheme for grouping tracks into object clusters within a shot. This section explains an object mining module for grouping different instances of the same object from different clusters into a single object group. Input to the module is the object clusters from all the shots in the video. Each object cluster contains a number of tracks. A track itself is described by the average descriptor of LIRs in it. A similarity value between two object clusters is obtained as the percentage of matched tracks.

Consider two object clusters \mathcal{X} and \mathcal{Y} with N_x and N_y tracks. First an optimal one-to-one matching between the tracks are obtained by minimizing the following cost function using the Hungarian algorithm.

$$C(\mathcal{X}, \mathcal{Y}) = \sum_t C(\mathbf{X}_t, \mathbf{Y}_{\phi(t)}) \quad (5)$$

where, the individual matching cost between the two tracks, $C(\mathbf{X}_t, \mathbf{Y}_{\phi(t)})$, is the L2 distance between the average SIFT descriptors of the tracks \mathbf{X}_t and \mathbf{Y}_t . The matching between the two set of tracks is further pruned by only allowing matches with the individual matching cost less than the threshold value τ_s . Let N_{xy} be the number of matched tracks. If the matching ratio, $R(x, y)$, defined as

$$R(x, y) = \frac{N_{xy}}{\min(N_x, N_y)}, \quad (6)$$

is higher than a threshold value, τ_{ob} then the object clusters \mathcal{X} and \mathcal{Y} are defined as belonging to the same object group. Since the number of tracks in each object cluster vary, match ratio is a more appropriate measure of similarity than the absolute number of matched tracks.

However, simple grouping of object clusters may not be sufficient to identify same objects due to large appearance variation and partial occlusion of objects in shots. Therefore we devised an algorithm to group the different instances of an object into same cluster. Formerly the algorithm can be easily explained using a graph. Let each object cluster be a vertex in a graph. The edge between the object cluster V_i and V_j is defined if the two clusters are matched.

$$E_{ij} = \begin{cases} 1 & \text{if } R(i, j) > \tau_{ob} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The graph will contain a number of connected components which are completely disconnected with respect to each other. Each connected component will correspond to an object group. Standard graph algorithms are used to calculate the number of connected components as well as to identify the individual object clusters of each object group. The algorithm is illustrated on an example in Fig 1. If $\tau_{ob} = 1$, each of the vertex will form a separate connected component. If $\tau_{ob} = 0$, all the vertices will join to form one single connected component. If $0 < \tau_{ob} < 1$, vertices will be grouped into a number of separate connected components. A smaller τ_{ob} value (0.1) is used in the mining experiments.

It is interesting to note that the same category objects are joined into a single group with a lower τ_{ob} value. For example, different faces are joined to form a group, different name boards are brought together into a single group and different vehicles are joined to one group. This is because they have similar features (features from eyes, nose and mouth for the face group and features from same letters for name board group). This is shown in Fig 1(d). Therefore with the aid of a classification algorithm, these phenomenon can be exploited in classification of object groups in videos.

3. RESULTS

The proposed algorithm is experimented with a full length movie *Groundhog Day*. The movie is divided into shots and

features are extracted from stable tracks as explained in the proposed approach. These shot features are grouped into clusters. The mining module mined the different instances of the objects from different clusters into relevant groups. Any object is considered as a valid mined object only if it is mined in at least five different shots. Under this condition 147 objects are mined from the whole movie. Some of the mined objects and the number of clusters they are mined are given in Table 1. Since we mine only one cluster from a shot into any particular group, the number of shots in which an object mined will be equal to the number of clusters mined for that object. Table 1 also gives number of missed and false clusters and precision, recall values for the given mined objects. The ground truth positions are manually obtained. The face of the main actor of the movie is mined in 81 different shots. The face of another actress is mined in 71 different shot. Despite the difference in scaling and viewing angle, these different instances of the faces are mined into relevant groups. It is worth noting that the system is able to mine small objects (eg microphone) as well as big objects (eg front of the red vehicle) with a single set of parameters.





















4. CONCLUSIONS AND FUTURE WORK

A novel framework for automatic object mining based on local invariant region descriptors is proposed. An algorithm previously published by the authors is used to divide the video into meaningful shots and extract representative features from stable tracks. These stable tracks are grouped into meaningful object clusters. These object clusters may contain a number of similar instances of the same object, and these instances are grouped together in the mining stage. The performance is evaluated with a full length movie, and excellent results are shown. Future work will consider classifying video objects using the clustered features.

5. REFERENCES

- [1] Brachman Ronald J, Khabaza Tom, Kloesgen Willi, Piatetsky-Shapiro Gregory, and Simoudis Evangelos, "Mining business databases," *Communications of the ACM*, vol. 39, pp. 42–48, 1999.
- [2] Etzioni Oren, "The world-wide web: quagmire or gold mine?," *Communications of the ACM*, vol. 39, pp. 65–68, 1999.
- [3] Josef Sivic and Andrew Zisserman, "Efficient visual content retrieval and mining in videos," *In Proc. Pacific-Rim Conference on Multimedia*, 2004.
- [4] Josef Sivic and Andrew Zisserman, "Video data mining using configurations of viewpoint invariant regions," *In Proc. CVPR*, 2004.
- [5] A. Anjulan and N. Canagarajah, "Video scene retrieval based on local region features," *In Proc. ICIP*, 2006.
- [6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," *In Proc. BMVC*, p. 2002, 384-393.
- [7] D.G. Lowe, "Distinctive image features from scale-invariant key points," *Int. Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

Table 1. Mined Objects The table shows results for some of the mined objects. Correct-Number of correctly mined clusters, Miss-Number of Missed clusters, False-Number of wrongly mined clusters.

Mined Object	Correct	Miss	False	Precision	Recall
	81	292	1	0.988	0.217
	70	131	1	0.986	0.348
	6	4	0	1.000	0.600
	5	2	0	1.000	0.714
	13	1	0	1.000	0.929
	5	1	0	1.000	0.833
	5	0	0	1.000	1.000
	5	0	0	1.000	1.000
	5	18	0	1.000	0.217
	6	6	0	1.000	0.500
	23	3	0	1.000	0.885
	8	4	1	0.889	0.500
	9	4	0	1.000	0.692
	7	7	0	1.000	0.500
	6	6	0	1.000	0.500
	6	2	0	1.000	0.750
	8	3	0	1.000	0.727
	5	1	0	1.000	0.833
	5	5	0	1.000	0.500
	10	3	0	1.000	0.769