

Video Scene Segmentation with a Semantic Similarity

Niraj Kumar, Piyush Rai, Chandrika Pulla, C V Jawahar

Center for Visual Information Technology
IIT Hyderabad
Gachibowli, Hyderabad-32
India

Abstract. Video Scene Segmentation is an important problem in computer vision as it helps in efficient storage, indexing and retrieval of videos. Significant amount of work has been done in this area in the form of shot segmentation techniques and they often give reasonably good results. However, shots are not of much importance for the semantic analysis of the videos. For semantic and meaningful analysis of the videos (e.g. movies), scene is more important since it captures one complete unit of action. People have tried different approaches in scene segmentation but almost all of them use color, texture etc. to compute scene boundaries. In this paper, we propose a new algorithm based on a Bag of Words(BoW) representation which computes semantic similarity between shots using a Bipartite Graph Model (BGM). Based on semantic similarity, we detect the scene boundaries in the movie. We have tested our algorithm on a multiple Hollywood movies, and the proposed method is found to give good results.

1 Introduction

Efficient and effective management of large amount of visual data is an important challenge [18]. Management of videographic data requires effective methods to process, organize, summarize and index in a semantically meaningful manner. Often video shot is taken as a fundamental unit to process the videos. Detection of shots can now be done relatively reliably for a variety of transitions [19][15]. However, for many of the high level video processing tasks, we need a scene level description of the video. A video scene is typically defined as a series of shots constituting a unit of continuous related concept (such as a fixed setting or the same action). Notion of scene is far more semantic in nature, compared to that of shots. Video scene segmentation refers to the partitioning of a video into a continuous sequence of shots or frames that are homogeneous in a semantic manner. It is a fundamental problem in semantic video processing and its solutions have many applications in video summarization, indexing, and retrieval [20][16][17].

There have been some previous attempts for scene segmentation in the last decade. Rasheed *et al.* [1] proposed a two-pass algorithm to segment the movie/TV shows into scenes. In the first pass, the scene boundaries are detected based on color similarity feature among the shots present in a window (Backward Shot Coherence). In the second pass, the over segmented scenes are merged based on motion similarity constraint. In [2], a weighted undirected graph called Shot Similarity Graph (SSG) is first constructed. The similarities between the nodes (weights in the graph) of SSG is computed

using both color and motion information. Then the SSG is split into smaller story units by applying the normalized-cut [8] technique for graph partitioning. This is a divisive method to segment the movie into scenes unlike the previous method. Zhai *et al.* [3] proposed a temporal video segmentation, where an arbitrary number of scene boundaries are randomly initialized and they are automatically updated using two types of updates — diffusion and jump. The updates of the model parameters are controlled by a Markov Chain Monte Carlo (MCMC) process. Most of these methods are based on either local or global constraints. Zhiwei Gu *et al.* [4], states that not only the global distribution of time and content, but also the local temporal continuity should be taken into account. They propose an Energy Maximization based Segmentation(EMS), in which the global and local constraints are represented by content and context energy. These energies are optimized in two steps. First the content energy are modeled by fitting generative model (using EM) to estimate the initial values of scene labels. And then the iterative conditional modes (ICM) are used for context energy to find global optimization.

Efficient video segmentation has traditionally relied on a proper selection of features [21][22] and an appropriate distance measure [23]. Examples of popular features include color, texture, motion vectors etc. Different features and homogeneity criteria generally lead to different segmentation of the same video. Often, the segmentations thus obtained are at the level of shots. Similarity across shots is used for scene segmentation.

Previous approaches [1][2] uses color as a feature to compute similarity between shots and consider only pairwise similarity that is single length path similarity to define the similarity between shots. The first problem with this approach is that the color does not take into account the semantics of the shots. The second problem arises with pairwise similarity. For a set of given three shots a , b and c , it is possible that similarity between a and b is lesser than what we get by adding similarities between a - c and c - b . Hence, it is possible that multiple path length similarity is greater than the direct path similarity or single path length similarity. Thus, to get a better similarity matrix for video shots, we need to consider multiple length path as well, instead of just a single length path. This gives us the semantic similarity between shots and this should give the better result than the previous work and approaches.

Determining semantic similarity between two sets of words is an important problem also in text domain [24]. It is often addressed with the help of Bag of Words Model. Documents (web pages etc) are represented in the form of bags-of-words (BoW) model. The idea is that each document is associated with tags (generally some keywords) and using those tags we compute the semantic similarity between two web pages based on concept similarity of the words occurring in the documents. But employing the same method in the images/videos is considerably difficult. The main problem is that how we associate textual tags with images/shots. This may require high level understanding of the image/video. Rather, we use a visual-bag-of-words model [10] for the videos. The problem at this step is that we cannot directly get the semantic meaning behind the visual words as they are only some feature points unlike the text domain where the tags itself can give some semantics to the web pages. Therefore, it is important to devise a way in which the semantic similarity of two shots in a video can be computed.

In this paper, we propose a method to compute semantic similarity between shots and then using that to get the scene boundaries of the movie. Section 2 describes the basic method. Section 3 presents details of the semantic similarity computation with the help of a bipartite graph model. Section 4 contains various experiments and results to validate the utility of the proposed methods.

2 Scene Segmentation with a Similarity Matrix

Scene is a set of contiguous shots which are connected by a central concept or theme. First we find shots and key frames in the given movie and then we propose a *bag-of-visual-words* (BoW) model representation of shots. Followed by this, we compute a semantic similarity matrix which gives the semantic similarity between each pair of shots. Then we apply normalized cut on the graph induced by the similarity matrix to group the shots into scenes. Figure 1 depicts the general flow of the scene segmentation algorithm proposed.



Fig. 1. This figure depicts a schematic flow of how the scenes are being detected in the movie. Initially we have the complete movie as input. We apply *shot segmentation* and *key frame extraction* to get the shots and key frames. Then *Semantic Similarity Computation* module computes semantic similarity between shots. The *Semantic Similarity Score* or *SS Score* has been shown in the figure for two pairs of shots. Then *Scene Boundary detection* module takes the shots and similarity score as input and output the scene boundaries.

2.1 Shot Segmentation and Key frame Extraction

Shot segmentation and *key frame extractions* has been studied widely and a number of approaches are available in this area [7][6]. To segment the *movie* into *shots*, we

use the method described in [7]. We take color histogram of consecutive frames and if difference between the histograms is above a certain threshold, the former frame is declared as shot boundary. After getting the *shots* in the movie, we use a variant of the unsupervised clustering method as described in [6] to get the key frames from each of the shots. Given a shot, we represent each of the frames in the shot as histogram taken in HSV space. Then we apply clustering on those histogram points to get the most suitable histogram points (cluster centers) and those frames corresponding to the cluster centers are taken as the key frame for that shot. For simplicity, we have predefined the number of clusters and so number of key frames extracted from each of the shots is same.

2.2 Semantic Similarity

Semantic Similarity between a pair of shots is defined as the similarity between them based on visual content of those shots. We define '*Semantic similarity*' between two shots based on following two conditions:

- If an object 'o' appears in shot S_i and the same object appears in shot S_j , then there exist a similarity between S_i and S_j based on object 'o'.
- If shot S_i is similar to shot S_j and shots S_j is similar to shot S_k then S_i is also similar to S_k . This type of similarity is called 'Transitive Similarity'.

We have defined a *scene* as a set of shots which are continuous in time and which are connected via a central concept (for example, a conversation). A conversation scene can be of type ABAB... or ABCABC.... where former represents a scene in which two persons are talking consecutively one by one and the later represents a scene where three persons are talking one after the other. Now looking at the content of these shots when we try to see the difference between these shots what we find is that the difference among these shots exist because of foreground objects e.g face of the person while most of the background or environment object (e.g wall, table etc) remains the same. Now because of our first hypothesis, there will be similarity between shots based on background objects or environment objects. Now to understand the relevance of the second hypothesis, consider a car racing scene or a scene of the highway. if there are three consecutive shots such as A[ab]B[bc]C[cde] where A, B and C are the shots and a,b,c,d,e are the content or objects or vehicle in the shots. In this case, there will be a similarity between A and B based on object 'b' and further there will be a similarity between B and C based on object 'c' using the first hypothesis. On the other hand, there will be a similarity between A and C because A is similar to B and B is similar to C according to the second hypothesis. The above two examples explains why the meaning or semantics of the scene will be captured in similarity computation using the two hypothesis given above.

2.3 Semantic Similarity Matrix Computation

In this section, we explain an algorithm which captures the two hypothesis given in the previous part approximately to compute the semantic similarity between shots. To achieve this, we first compute *Bag-of-VisualWords* representation for the shots of the

movie. The set of all the visual words for a movie is the visual vocabulary for that movie. Visual words are computed by taking all the keyframes from all the shots and applying clustering on the SIFT features extracted from those keyframes. Now a shot S_i can be represented as a K dimension vector (histogram) where S_{ij} gives the count of j^{th} visual word in i^{th} shot and K is the total number of visual words or vocabulary. A visual word w_j appears in shot S_i if there is some feature point from keyframes of i^{th} shot which lies in j^{th} cluster where j^{th} cluster represents j^{th} visual word. Using this convention, a shot is represented by

$$S = n_1, n_2, \dots, n_j, \dots, n_K \quad (1)$$

where n_j gives the frequency of j^{th} visual word divided by total number of visual words present in this shot, and K is the size of the vocabulary. With this representation of the shot, we use Euclidean distance and intersection distance to find the distance between pair of shots. Euclidean distance between i^{th} and j^{th} shot is given by $D_{ij} = \sum_{k=1}^K |n_{ik} - n_{jk}|^2$ and intersection distance between i^{th} and j^{th} shot is given by $D_{ij} = \sum_{k=1}^K \min(n_{ik}, n_{jk})$. Here n_{ik} gives the normalized frequency of k^{th} visual word in i^{th} shot. The similarity between the shots will be inversely related to this distance. Once we compute all pairwise distance between two shots, we can visualize a graph $G(V, E)$ where V is the set of vertices consisting of all the shots and E is the set of edges between each pair of shot and each edge joining i^{th} and j^{th} shot is associated with an edge weight equal to distance computed between i^{th} and j^{th} shot as explained above.

Now, We define a matrix D^l of size $t \times t$ where t is number of shots. D_{ij}^l is defined as the minimum distance between i^{th} and j^{th} shot by considering path length up to l . Path length tells the maximum number of vertices(shots) used in computing the minimum distance between two shots. The value l is decided based on expected number of shots in a scene. We can compute D^l matrix using either the straightforward matrix multiplication algorithm which takes $O(t^4)$ time or Floyd Warshall algorithm [12] which takes $O(t^3)$ time. After applying this algorithm, we get a similarity matrix such as

$$Sim_{ij} = \frac{1}{D_{ij}^l} \quad (2)$$

2.4 Scene Boundary Detection

Rasheed and Shah [2] have used normalized cut [8] technique to partition their shot similarity graph into scenes. We also use a similar approach. After finding the similarity between shots, our aim is to partition the shots into scenes in a way such that

- All the shots in a scene are continuous in time.
- Intra similarity within a scene is maximized.
- Inter similarity between scenes is minimized.

To do this, we construct a graph $G(V, E)$ where V is the set of all the shots and E is the set of edges joining each pair of shots. e_{ij} is the edge joining i^{th} shot with j^{th}

shot. Each edge e_{ij} is associated with a weight w_{ij} such that

$$w_{ij} = f(i, j) \cdot Sim(i, j) \quad (3)$$

Here $f(i, j)$ accounts for *temporal similarity* between i^{th} and j^{th} shot and $Sim(i, j)$ is the *semantic similarity* between i^{th} and j^{th} shot. $f(i, j)$ term is important because there might be a case when two shots which are far apart are visually similar but the similarity between them should decrease because of the distance. We compute $f(i, j)$ as inverse of the absolute difference between i and j .

Normalized cut as described in the paper [8] is used to *partition* this graph $G(V, E)$ into two parts $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ such that $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \phi$. The idea behind this algorithm is to maximize *intra class similarity* and minimize *inter class similarities* which is also our objective. However, if we apply normalized cut directly on the *graph*, we may find some *partitions* of the graph where *shots* will not be *contiguous* in time but we need to *partition* the graph in such a way that shots are *contiguous* in time i.e., for each graph $G(V, E)$ which is partitioned into two sub-graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, the following condition should hold

$$(i < j \text{ or } i > j); \forall v_i \in V_1, v_j \in V_2 \quad (4)$$

We use the objective function given in Equation 5 to find the weakest link between two consecutive shots in the graph and then apply the same procedure recursively on either side of the partition.

$$Ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{Assoc(V_1, V)} + \frac{cut(V_2, V)}{Assoc(V_2, V)} \quad (5)$$

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij} \quad (6)$$

$$Assoc(V^1, V) = \sum_{i \in V_1, j \in V} w_{ij} \quad (7)$$

Using the above equations, we compute $Ncut$ values for each of the *probable partition* (for example initially there are $n - 1$ possible partition) and find the *partition* for which $ncut$ value is *maximum* and if this $ncut$ value is above a *threshold*, we divide the *graph* into two *partitions* and the same procedure is repeated recursively on the subgraphs.

3 BGM and Semantic Similarity Computation

3.1 Bipartite Graph Model(BGM)

The algorithm proposed in the previous section shows some improvement in result when compared with [2]. This could be due to the better features/representation we use compared to the previous method. However, it does not capture the full semantics of a

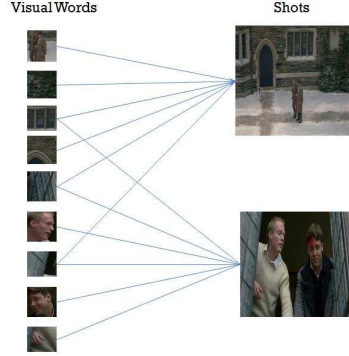


Fig. 2. This figure shows a bipartite graph model where all the visual words are on the left hand side and all the shot nodes are on the right hand side.

scene and is also an inefficient algorithm. To overcome these issues, we look in the text domain about how they solve the problem of computing semantic similarity between documents. One interesting idea is to use Bipartite Graph Model to compute semantic similarity between documents. The approach uses *tf-idf* values to build a bipartite graph and then using this graph, they compute semantic similarity. One similar approach in image domain is image retrieval application [5] that uses Bipartite Graph Model to retrieve the similar images efficiently. We have tried to use a near similar idea in video domain to compute semantic similarity between shots. We have already represented shots in terms of *visual words*. We define two parameters-*shot-visual word*, *SV* matrix which is equivalent to *term-document* matrix and *inverse shot frequency*, *ISF*, for the visual words which is equivalent to *inverse document frequency* in text domain. *SV* and *ISF* is given by:

$$SV(i, j) = \frac{\text{count of } j^{\text{th}} \text{ visual word in } i^{\text{th}} \text{ shot}}{\text{total number of visual words in } i^{\text{th}} \text{ shot}} \quad (8)$$

$$ISF(i) = \frac{\text{count of shots in which } i^{\text{th}} \text{ visual word appears}}{\text{total number of shots}} \quad (9)$$

Then we construct a *bipartite graph* as shown in Figure 2. All the visual words are on the left side of the graph while on the right side are all the shot node. A bipartite graph with visual words, shots and edges can be represented as $G(W, E, S)$ where W is the set of all visual words, $W = w_1, w_2, \dots, w_k$; S is the set of all the shots and is given by $S = s_1, s_2, \dots, s_n$ and E is the set of all edges and $E = e_{s_1}^{w_1}, e_{s_2}^{w_2}, \dots, e_{s_n}^{w_k}$. Here $e_{s_i}^{w_j} = SV(i, j)$. Besides this with each visual word, is associated *ISF* for that *visual word*. An edge joining j^{th} visual word with i^{th} shot has an edge weight which is normalized frequency of the j^{th} visual word in the i^{th} shot.

3.2 Semantic Similarity Computation Using Bipartite Graph

Using this bipartite graph, we can also compute the intersection distance matrix D which has been computed in section 2.3. Intersection distance between two shots is given by:

$$D(i, j) = \sum_{k=1}^K \min(s_{ik}, s_{jk}) \quad (10)$$

We can compute the same thing using this graph as

$$D(i, j) = \sum_{k=\text{index of common visual words}} \min(e^{ik}, e^{jk}) \quad (11)$$

Here e^{ik} is the actual frequency of k^{th} visual word in i^{th} shot. However for all the subsequent discussion, weight associated with e^{ik} is the normalized frequency of k^{th} visual word in i^{th} shot. The above equation does not include transitive similarity and computationally also it is not better than the previous one discussed in section 2.4. Therefore, instead of computing like this, to find the similarity of i^{th} shot with all other shot, i^{th} shot node is given some initial cash to distribute among other shot nodes in BGM based on relevancy. The relevancy is decided based on the edge weights joining visual word nodes with shot nodes. The propagation of cash through BGM continues until it runs out. The higher the amount of cash flowing through a shot node, the higher is the similarity of that shot with i^{th} shot. This algorithm is applied in such a way that a shot node is given some cash initially. Now if this node is a shot node, it will propagate the cash to the ‘visual word nodes’ in the proportion of edge weights joining shot nodes with the visual word nodes. If the cash is propagated from a visual word node, some part of the cash is absorbed at the word node in the proportion of ISF of that visual word and rest amount of cash is propagated to the shot nodes joined with this visual word. If the cash flowing through BGM goes below a certain threshold, we stop cash propagation. At this point, all the shot nodes have the amount of cash they received. For a j^{th} shot node total cash received is

$$cash_{total}^j = cash_{previous}^j + cash_{current}^j \quad (12)$$

We take the cash values at different shot nodes as the similarity between i^{th} and j^{th} node i.e $\text{Sim}(i, j) = cash_{total}^j$, when i^{th} shot node was given the initial cash. We apply this algorithm τ times for each of the shot node to calculate all pairwise semantic similarity where τ is equal to number of the shots.

Absorption of amount value at the visual word nodes is very crucial to make the algorithm more optimized with respect to time and space as it prunes the tree and so does not apply this algorithm for the frequent occurring visual words. The more a visual word is frequent, the less useful it is for the purpose of making distinction between shots and so less useful in establishing semantic similarity. Also this algorithm models and captures both of the hypothesis given in section 2.1. It will ensure that there will be a semantic similarity between shots if an object ‘o’ is common to them because in that case some amount will flow from one side to the other side increasing

its semantic similarity. Also it will capture transitive similarity of more than two path lengths and the maximum length of transitive path will depend on threshold.

Using the above procedure, we compute Semantic Similarity matrix which contains pair wise semantic similarity between each pair of shots. Then, we use the approach described in section 2.4 to find the scene boundaries or to group the shots into scenes.

4 Results

We evaluate the performance of our approach on three different videos. The first video is 36 minutes long, taken from the movie “A beautiful Mind”, the second video set is 51 minutes long taken from the movie “The Fellowship of the ring” and the third video 60 minutes long taken from the movie “Gladiator”. The three movies are of different genre .The first is a drama/history slow paced movie while the second is an action/adventure/fantasy movie and the third is an action/adventure/drama. For testing the performance of our algorithm we need a ground truth scene which were manually detected. In the phase of computing the bag of words we took about 700,000-800,000 sift features in total and then clustered them assuming 1500 centers i.e. the visual words. The results obtained using our algorithm are compared with the results of the method proposed in [2]. For both the approach we used the same videos mentioned above. The parameters used for performance evaluation are Recall : CT/GT, Precision: CT/DT,

Table 1. Comparison of the previous work results and results obtained by proposed method

Movie	Rasheed and Shah’s approach			Proposed Method		
	BM	LOTR-1	GD	BM	LOTR-1	GD
#Shots	219	358	363	219	358	363
Duration (min)	36	51	60	36	51	60
#G. Truth Scenes	18	29	28	18	29	28
#Detected Scene	28	35	43	23	33	42
#Correct Scene	15	21	25	16	23	26
#False Negative	3	8	3	2	6	2
#False Positive	13	14	18	7	10	16
Recall	0.833	0.724	0.893	0.889	0.793	0.929
Precision	0.536	0.60	0.581	0.696	0.697	0.619
F-Score	0.652	0.656	0.704	0.781	0.742	0.743

F-score: $\frac{2*precision*recall}{precision+recall}$, where GT is Ground Truth scene boundaries, CT is Correctly Detected scenes boundaries, DT is Detected Total scenes boundaries.

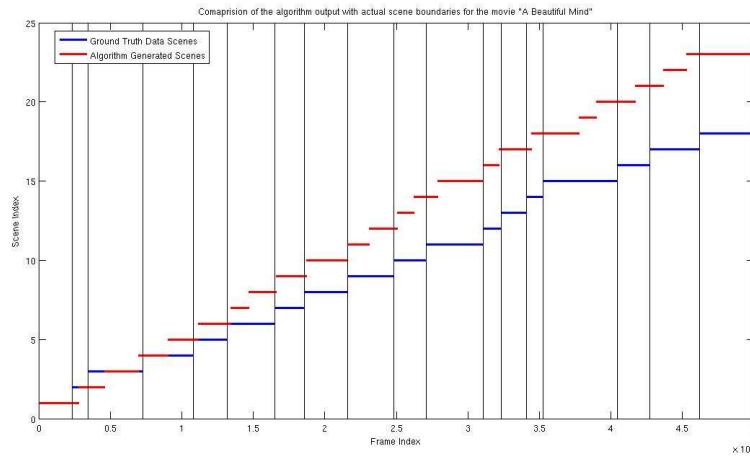
The output scene boundaries from the algorithm are compared with the ground-truth scene boundaries such that a 30-seconds sliding window was swept over the detected boundaries as tolerance factor. Finally, F-score is used to compare the results of the algorithm proposed here and the algorithm mentioned in [2]. For proper comparison we implemented the method proposed by Rasheed *et al.* [2] and the Table-1 shows comparative results of both the algorithms.

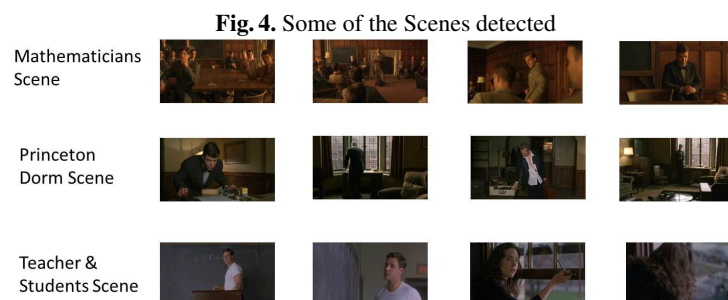
We have selected SIFT features for the representation of the video. But there can be some other simple features as well. One such feature can be to use color histogram to represent the shots of the video. The color histogram is calculated by dividing the key frames into fixed size patches and then using hsv bins to represent the shot by color histogram. We compared the algorithm accuracy when the color histogram is used as the feature and when sift features used to detect key-points and the results are present in Table-2. We can clearly see sift feature in general gives better results compared to other simple representation of the shots.

Table 2. Comparison of the results when sift features and color histogram were taken as the features of the bag of words computation on the movie “A Beautiful Mind”

	Color Histogram as feature	Sift Features
# G. Truth Scenes	18	18
# Detected Scene	24	23
# Correct Scene	14	16
#False Negative	4	2
#False Positive	10	7
Recall	0.77	0.889
Precision	0.58	0.696
F-score	0.66	0.78

Fig. 3. Ground Truth data scenes and Algorithm detected scenes





In Fig. 3 we gave the visual representation of actual detected scene when compared to the ground-truth scene boundaries. In Fig. 4 we have shown some of the detected scenes in the movie “A Beautiful Mind”.

5 Conclusion

We present a method of partitioning a video , particularly movie, into scenes. We have considered the fact that a scene consists of shots which are semantically related and continuous in time. To achieve that, we have presented a method to compute semantic similarity between the shots. Once we get semantic similarity, we construct a graph and the problem transforms into graph partitioning problem. The approach presented here is found to be better than previous approaches in which semantic similarity between shots was not considered. It can be seen from the results that it is important to compute the similarity between the shots not only on the basis of direct similarity but it is important to consider transitive similarity as well.

References

1. Zeeshan Rasheed and Mubarak Shah : Scene Detection In Hollywood Movies and TV Shows, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA(2003)
2. Zeeshan Rasheed and Mubarak Shah : A graph theoretic approach for scene detection in produced videos, Multimedia Information Retrieval Workshop 2003 in conjunction with the 26th annual ACM SIGIR Conference on Information Retrieval
3. Yun Zhai and Mubarak Shah : A General Framework for Temporal Video Scene Segmentation In IEEE International Conference on Computer Vision, Los Alamitos, CA, USA(2005)
4. Zhiwei Gu and Tao Mei and Xian-Sheng Hua and Xiuqing Wu and Shipeng Li : Energy Minimization Based Video Scene Segmentation In ICME(2007)
5. Karthik, S. and Pulla, C. and Jawahar, C.V. : Incremental on-line semantic indexing for image retrieval in dynamic databases In Computer Vision and Pattern Recognition Workshops, 2009.
6. Yueting Zhuang and Yong Rui and Huang, T.S. and Mehrotra, S. : Adaptive key frame extraction using unsupervised clustering In ICIP 98
7. Zhang, HongJiang and Kankanhalli, Atreyi and Smoliar, Stephen W. : Automatic partitioning of full-motion video, 1993

8. Jianbo Shi and Jitendra Malik : Normalized Cuts and Image Segmentation In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000
9. Josef Sivic and Bryan C. Russell and Alexei A. Efros and Andrew Zisserman and William T. Freeman : Discovering objects and their location in images In IEEE Intl. Conf. on Computer Vision, 2005
10. Sivic, J. and Zisserman, A. : Video Google: a text retrieval approach to object matching in videos In Ninth IEEE International Conference on Computer Vision, 2003.
11. J. Zhang and S. Lazebnik and C. Schmid : Local features and kernels for classification of texture and object categories: a comprehensive study, International Journal of Computer Vision, 2007
12. Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein : Introduction to Algorithms, second edition, 2001
13. Lowe, David G. : Object Recognition from Local Scale-Invariant Features In ICCV'99.
14. Baeza-Yates, Ricardo A. and Ribeiro-Neto, Berthier : Modern Information Retrieval, 1999.
15. Rainer Lienhart : Comparison of automatic shot boundary detection algorithms, 1999.
16. A. Hampapur : Virage video engine In SPIE, 1997.
17. D. DeMenthon : Relevance Ranking of Video Data using HMM Distances and Polygon Simplification In Int. Conf. on Adv. in Visual Info. Systems, 2000.
18. Wactlar, H. : The Challenges of Continuous Capture, Contemporaneous Analysis, and Customized Summarization of Video Content In Defining a Motion Imagery Research and Development Program Workshop, 2001.
19. Zhang, HongJiang and Kankanhalli, Atreyi and Smoliar, Stephen W. : Automatic partitioning of full-motion video In Multimedia Syst, Springer-Verlag New York, Inc, 1993.
20. R. Smith. : VideoZoom spatio-temporal video browser In IEEE Tran. on Multimedia, 1999.
21. Thomas Deselaers and Daniel Keysers and Hermann Ney : Abstract Features for Image Retrieval: An Experimental Comparison, 2007 .
22. Thomas Deselaers and Daniel Keysers and Hermann Ney : Features for Image Retrieval: A Quantitative Comparison, 2004
23. A Vadivel and A K Majumdar and Shamik Sural : Performance comparison of distance metrics in content-based Image retrieval applications, 2004 .
24. Danushka Bollegala, Yutaka Matsuo, Mitsuru Ishizuka : Measuring Semantic Similarity between Words Using Web Search Engines, 2007 .
25. Ioannis Antonellis and Efstratios Gallopoulos and I. Antonellis and E. Gallopoulos : Exploring Term Document Matrices From Matrix Models in Text Mining, 2006