Jestr Journal of Engineering Science and Technology Review 8 (5) (2015) 12-18

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Research Article

Scene Retrieval in Videos by Query Image

G. Szűcs*

Dept. of Telecommunications and Media Informatics, BME, Hungary and Inter-University Centre for Telecommunications and Informatics,H-4028 Kassai út 26., Debrecen, Hungary

Received 23 April 2015; Accepted 30 November 2015

Abstract

In this paper the concept and details of an elaborated video scene retrieval system is described. The videos can be segmented into scenes, and these semantic parts of video are the available set for the end user and in our retrieval system. The end user gives a query image, and would like to retrieve the most relevant scenes from a long video, furthermore the user may wonder the results in more details – in less unit than scene, e.g. in sample image – ordered as well. The key problem of such retrieval is the speed of the answer, and the solution for fast search is described in this paper. An index structure, as contribution of this paper is outlined for speed up; but for building this structure a clustering phase should has been applied before. After feature extraction from query image and comparison with index structure the candidate images for relevant hits are available; however many irrelevant images also can be found in the hit list, which causes difficulty. The paper shows a solution for filtering the hits, and finally the images will be in descending order based on relevance. At the end of the paper a solution has been shown for ranking of the scenes based on the rank of the images in the hit list.

Keywords: clustering, index structure, ranking, multimedia retrieval, video scene

2. Introduction

Image and video retrieval has been an active research topic in recent years due to its potentially large impact on both image and video understanding and Web image search. Comparing two types of image/video retrieval techniques, i.e. annotation based retrieval and content-based image/video retrieval it can be stated that there is a growing interest in content-based retrieval because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image/video retrieval 11.

Content-based retrieval allows finding information by searching its content rather than its attributes. In this area content-based video systems can accurately and automatically process huge amounts of videos. These systems require in their first stage to segment the video stream into separate shots. Afterwards features are extracted for video segments representation. Based on the representation a similarity/distance metric is chosen, and an algorithm retrieves the query, and related videos results will be the answer. This paper focuses on representation, and shows an efficient representation solution for quick response.

In this paper the problems of video scene retrieval are discussed, where scenes as segments of video are available for the system. The end user gives a query image, and would like to retrieve the most relevant scenes from a long video, furthermore the user may wonder the results in more details – in less unit than scene, e.g. in sample image – ordered as well. The query image can be a sample image from video, or part of this sample image, or another image from any other source. For example the end user gives an image of interesting subject as can be seen in a frame of the video, and the user would like to see the scenes in that this subject can be found. In a long video it can happened that many scenes contain this query image, in one of them the interesting subject can be seen with little size (e.g. as small subject in a corner), in another scene the interesting subject can be seen in total (where only the subject can be seen in the whole picture); the last case will be more relevant for the user, so the retrieval system should order the results. One of the key problems of such retrieval is the speed of the answer, and the solution for fast search is also presented in this paper.

One of the aims in video analysis is to group the shots into temporal scenes, such that all the shots in a single scene are related to a particular physical setting, an on-going action or a theme. This paper does not deal with video segmentation into scenes, in the literature there are some solutions 165 for this. A general framework has been developed for temporal scene segmentation using statistics and Markov chain Monte Carlo (MCMC) technique 15 to determine the boundaries between video scenes. There is another suggested system 6 for content based video retrieval by using adaptive threshold for video segmentation and key

* E-mail address: szucs@tmit.bme.hu
ISSN: 1791-2377

frame selection as well as using both low level features together with high level semantic object annotation for video representation.

After segmentation there are some possible tasks related to video scenes, these can be analyzed 7, can be classified into different genre, for example musical, horror, etc. A horror video scenes recognition algorithm 12 has been solved already by introducing color emotion and color harmony theories.

In paper 2 a retrieval system is presented, in which the users can choose keyframes which contain the interest items, and the system recommends frames with similar content to the target video. The paper is promising, but it does not deal with sub-regions of keyframes and scenes of video. Another work 9 has proposed fast image retrieval for large scale as well, but only for images, not for videos.

One of the most important issues in the retrieval is the indexing. A new concept 4 is multimodal video indexing, where joint probability density functions of audio and visual features has been used in order to fuse features from different modalities. This paper also focuses on video indexing, but based on only images, so first of all the image retrieval is discussed. Originally proposed for text retrieval using bag-of-words (BoW) models, the technique of inverted index can be also applied for image retrieval. Once the codebook is constructed from feature descriptors extracted from the image database, each image will be represented by a BoW model, which is a histogram representation and each attribute indicates the number of occurrences of each visual word. Once this BoW model is obtained for each image, one can consider each visual word as an entry of an inverted file, and this file records the list of images containing that visual word. If the inverted files and a query image are given, the results can be produced quickly based on them 1714. In the paper 10 the authors have applied region color features in constructing the codebook, which can be used in only when the color information does not have large variations. In our work described in this paper SURF 3 features have been used, which more complex, but can be used for large variations of color as well.

2. Concept of the scene retrieval by image query

The searching task is to seek appropriate video parts or scenes by a sample image and to play it for the user. The essential part of the task is the seeking the most similar image among the very large picture sample set of video. After finding the best image possessing point in time, the video can be played (i) from this point or (ii) from the beginning of the scene in which this point can be found. These two cases can be compared: at first case the viewer person will see the part of the video from the most important point, so he or she could not wait any second for the searched part. But in this case the prelude (precedents) of the played part is missing, which may be semantically important for the viewer. At the second case the user will see the whole scene, so the viewer will not omit important parts. This is appropriate for understanding, but a little disadvantage is the longer played part of the video. In our system the user can choose between the first case (video part) and the second case (video scene).

The difficulty of the seeking the most similar image is comparison of the query image with too many other images. The applied solution of this problem is inverted index, and the whole procedure of the scene searching can be seen in Fig 1. After the sampling of video the keypoints are extracted from each image. Based on clustering of keypoints an inverted index is constructed. The extracted keypoints of the query image are compared to the inverted index, and based on this the keypoints are filtered. The most similar images can be searched and after the choosing the appropriate video part or scene the system is able to play the video from the founded point for the viewer.

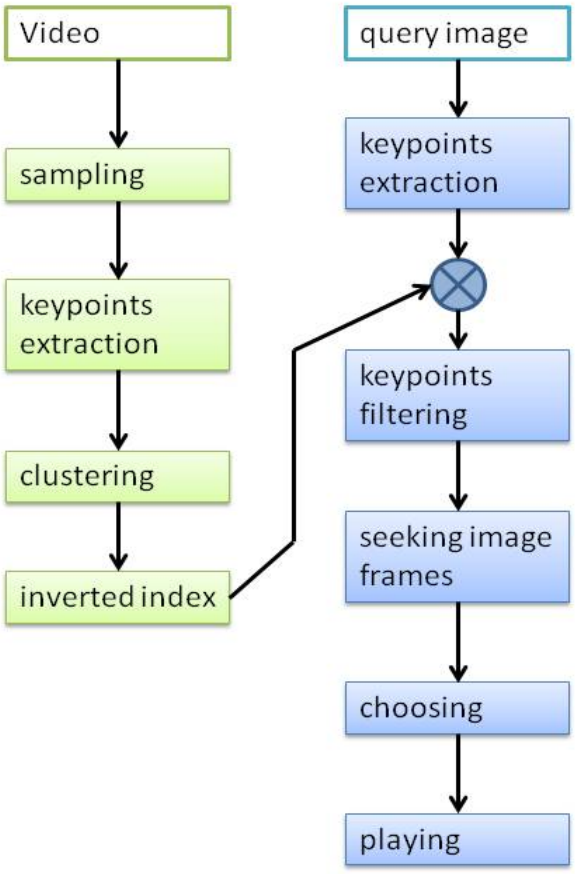

Fig. 1. Block diagram of the scene searching by sample image

2.1 Keypoint Extraction

The extracted keypoints and vectors, which belong to keypoints, come from SURF algorithm 3. Before running SURF algorithm, some preparation operations should be executed. One of them is the grayscale conversation, in which the colored image is transformed into grayscale one. Another preparation operation is the histogram equilibration. Using this operation the luminance of the picture can be normalized and the contrast can be increased.

The results of SURF algorithm are stored in HDF5 (Hierarchical Data Format) 23 format as can be seen in Fig. 2.

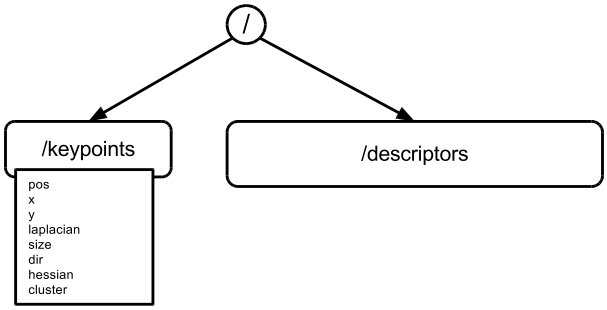

Fig. 2. Results stored in HDF5

The „/keypoints" is a table type element with the following items:

- pos: is the position (with given time code) of the sample image in the video, from which the keypoint is extracted.
- x: is the x coordinate of keypoint on the image.
- y: is the y coordinate of keypoint on the image.
- laplacian: is a sign of Laplace value (+1, 0, or -1).
- size: is the largeness of the keypoint feature.
- dir: direction of the keypoint feature (between 0 and 360).
- hessian: is based on the Hessian matrix.
- cluster: is the identity of cluster (explanation can be seen later).

All these values (except the *pos* and *cluster*) are return values of SURF, which supplies these for each keypoint extracted from images.

The */descriptors* is a matrix with 128 column and *N* rows, where *N* is the number of all keypoints on the given image of the video. This contains the description vectors (with 128 values) of the keypoints. There is a row in */keypoints* and a row with same index identity in */descriptors* for each keypoint.

2.2 Inverted Index Structure by Clustering

After the keypoint extraction on the query image the search task would be slow, because huge amount of comparisons should be executed. In order to increase the speed of execution for the end user the number of comparisons should be decreased; the system will be usable only in this case. Sampling an average film (second by second) the number of images will be 6000-7000, and number of SURF keypoints in an image is approximately 700-1000. Thus at case of comparison only one new keypoint with all others the order of magnitude is 10^6. Furthermore the number of keypoints in a query image is also more hundreds, so the number of total comparisons is hundred times larger than mentioned above. Notice, that in a comparison not only two values should be compared, but 128, because SURF features are represented as 128 dimensional vectors.

The majority of keypoints can be found in more images, because same objects can be seen during more than 1 second in the video. More precisely not the same keypoints occur in different images, but they are very close together, in consequence of little changing of objects in the film. The comparisons will be completed more times at same (or almost same) keypoint, which increases the searching time. If we could arrange the similar keypoints into one group, then number of operations during search time could be decreased. The clustering is a solution for build groups, in which the elements are similar to each other and elements in different groups are differ from each other.

The k-means method is a widely used clustering technique that seeks to minimize the average squared distance between points in the same cluster. Although it offers no accuracy guarantees, its simplicity and speed are very appealing in practice (it is standard practice to choose the initial centers uniformly at random from more dimensional space). By augmenting k-means with a simple, randomized seeding technique, a new algorithm, so called k-means++ [1] has been outlined with the optimal clustering. Preliminary experiments show that the augmentation improves both the speed and the accuracy of k-means.

The k-means algorithm begins with an arbitrary set of cluster centers, but k-means++ algorithm uses a specific way of choosing these centers. At any given time, let D(x) denote the shortest distance from a data point x to the closest center we hav e already chosen; so k-means++ algorithm is the following:

1a. Choose an initial center c_1 uniformly at random from X.

1b. Choose the next center c_i, selecting $c_i = x' \in X$ with probability p, where p can be calculated by Eq. 1.

$$p = \frac{D(x')^2}{\sum_{x \in X} D(x)^2} \tag{1}$$

1c. Repeat Step 1b until we have chosen a total of k centers.

2. For each $I \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in X that are closer to c_i than they are to c_j for all $j \neq i$.

3. For each $I \in \{1, \ldots, k\}$, set c_i to be the center of mass of all points in C_i, as can be seen in Equation 2, where ${}_m c_i$ and ${}_m x$ is the m^{th} coordinate of the c_i point and x point respectively.

$$ {}_m c_i = \frac{\sum_{x \in C_i} {}_m x}{|C_i|} \tag{2}$$

4. Repeat Steps 2 and 3 until C no longer changes.

Choosing the number of the clusters in k-means++ algorithm is a sensitive parameter for the goodness of the results. We have used the rule of thumb formulated in Equation 3 for the determination of the clusters.

$$k = \sqrt{n/2} \tag{3}$$

After the clustering an inverted index can be built from images and the clusters of keypoints. Inverted index shows the images in which keypoints in a cluster can be found. This index structure should be stored, so this is also a part of the preparation and it helps with small search time. So the previous structure described above is extended with new item: *cluster* (see Fig. 3). In this item a list is stored about the serial number (identity) of corresponding images.

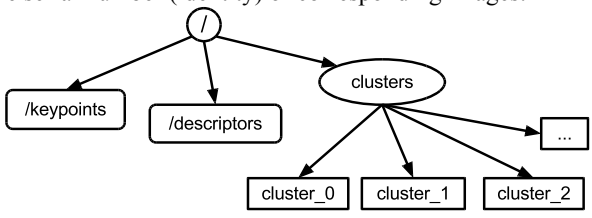

Fig. 3 Results stored with cluster information

The inverted index structure can be seen in Fig. 4, which is able to speed up the response at search phase by reading the inverted lists. In this structure each group (cluster)

contains the list of the image identities that possess at least 1 keypoint belonging to this cluster.

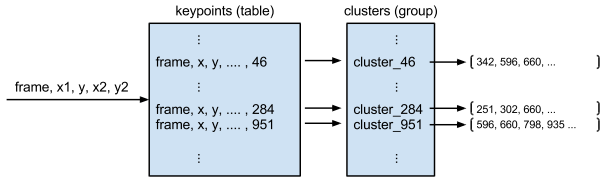

Fig. 4. Inverted index structure

2.3. Retrieval for query image

There are two possible way to give query image for end users:

- external image from anywhere
- internal image from selected region of a sample image of the video

At external case the unknown image should be preprocessed: keypoints are extracted from the image and stored during the search activity of the user. At internal case the preprocessing phase is omitted during the search activity, because keypoints are available, but the appropriate keypoints should be searched (filtered) among the large amount of keypoints. The end user selects a sample image in the video and selects a rectangle region on this image, so the serial number of the image and the coordinates of selected region are given. At */keypoints* table the values are filtered according to the condition of coordinates, i.e. $x_1 < x < x_2$ and $y_1 < y < y_2$, where x and y are the coordinates of extracted keypoint, $[x_1, y_1]$ $[x_2, y_2]$ are the coordinates of selected region on the image. The last field of the filtered keypoints (which are met with the conditions described above) is the identity of the cluster, thus the list of serial number of images related to this cluster is available at once.

The union of these lists gives the set of all images in the cluster related to keypoints in query image. Number of occurrences of an image should be summarized, because this can be larger than 1 in this set; moreover the more accurate a hit, the more times can be found in the set, because of more common keypoints. The order of the hits is based on this summarized value, the number of occurrences.

At internal query case the system has used the sample image of the video, thus computer time is saved during the search activity, because calculation of SURF keypoints on this image has not required, furthermore the information about each keypoint can be known that in which cluster can it be found. These advantages are due to preprocessing. At external case the task is little larger, because the keypoints should be extracted (as mentioned before) and the nearest cluster should be searched based on the distance of the keypoint and the centroid of the cluster. After these two steps the further phases are the same as described above.

The union list of hits is quite large, because this contains many hits possessing only very few common keypoints. Another problem may occur at clustering, because two (or more) keypoints can be got involved in same cluster because of similarity in spite of that, they are not related to images of relevant hits. The inadequate number of clusters increases the probability of such kind of mistakes.

2.4 Filtering the results and rank fusion

For decreasing the magnitude of union list of hits a method is introduced using the number of similar keypoints. At ordering of hits the number of occurrences of an image *I* in union list has been used, so this value is denoted by n_i. Let *N* be number of keypoints in the query image and the ratio of these values can indicate the goodness of hits (as can be seen in Eq. 4), so this would be helpful for choosing the relevant hits.

$$R = n_i / N \tag{4}$$

The value of *r* is between 0 and 1 for every sample picture, and the hits can be filtered by a predefined threshold, so if the *r* of an image is less than this threshold, then it will be filtered out. Only the rest (images with large *r*) will be processed further. During the development and testing it can be found that 0.6 is an appropriate value for the threshold. The goodness of these values depends on attributes of video and the number of clusters.

Let us consider an extreme case, where the number of clusters is maximal, i.e. this equals to all keypoints in all images. In this case if the query image is a part of a sample image from video (at internal query this is always true, at external query this can occur, but not always), then *r* is equal to maximal value, i.e. 1 for this sample image. However if consider clustering case, where the number of clusters is less than all keypoints in all images, then in a cluster more keypoints can be found. It can happened that some keypoints of the query image also get involved into common cluster, so the number of clusters to be joined at phase of union list will be less than the keypoints on the query image. (Note that lists contain the serial numbers of image, not the identity of keypoints.) In this case the maximal value of *r* is less than 1; naturally the image (as most relevant image) that contains the internal query image will reach this maximal value. Concluding the clustering can decrease the maximal value of *r*.

The ordered and filtered list of sample images is available with time code in the video. The system could offer the end user to play the video from this point of time, but usually the most relevant image does not appear very suddenly, so there are some previous and next images possessing more or less relevance. Additionally the context of the situation in a film is also important for end user to understand the part of the video. Therefore the playing should be started earlier, than the point of time of image in the hit list. One of the possible solutions is a fix time interval (e.g. 10 seconds) for beginning the video playing earlier. Better solution is to play the appropriate scene – in which the sample image can be seen – from the beginning because of whole context, since the minimal duration of video part for understanding is changing and a whole scene is sure enough to understand the context; but our system offers both of them.

A more sophisticated solution has been used in our system based on aggregation of images in the filtered hit list. The idea is the reciprocal rank fusion of rank of images as can be seen in Equation 5, where $rank_i$ is the rank of image *I* and *score_rank* is a real value (usually not integer) to final ordering.

$$score\_rank = \frac{1}{\sum_i \frac{1}{rank_i}} \tag{5}$$

The reciprocal rank fusion has been calculated for each scene (based on images that can be found in the scene) and

based on this rank (*score_rank*) the order of the scenes can be shown for the end user, where the scenes with smallest *score_rank* can be found in top of the hits. Finally these scenes can be played for the end user.

3. Implementation

This scene retrieval system has been implemented in a general-purpose, high-level programming language: Python 18. This programming language has been selected for this work, because interpreters are available for many operating systems, syntax is clear and expressive, furthermore free and open source modules (e.g. module for image handling 20) can be used in implementation. One of the open source modules is Open Source Computer Vision Library (OpenCV) 19 which a library of programming functions mainly aimed at computer vision. This is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface, e.g. there is now full interface in Python. So OpenCV third-party library (and extension of Python, so called NumPy) has been used in the solution described in this paper.

Another useful tool is Matplotlib 21, which has been used only in the planning of the system by drawing some mathematical functions. Another mathematical library is NumPy 22, which is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays. NumPy is open source (licensed under the BSD license, enabling reuse with few restrictions) and has many contributors. NumPy is a prerequisite of OpenCV, because mathematical operations are required for many image functions, so this was its role in our system.

Snapshot of the implemented system can be seen in the Fig. 5, where television was the internal query (green rectangle) and the results are presented at the bottom of the screen.

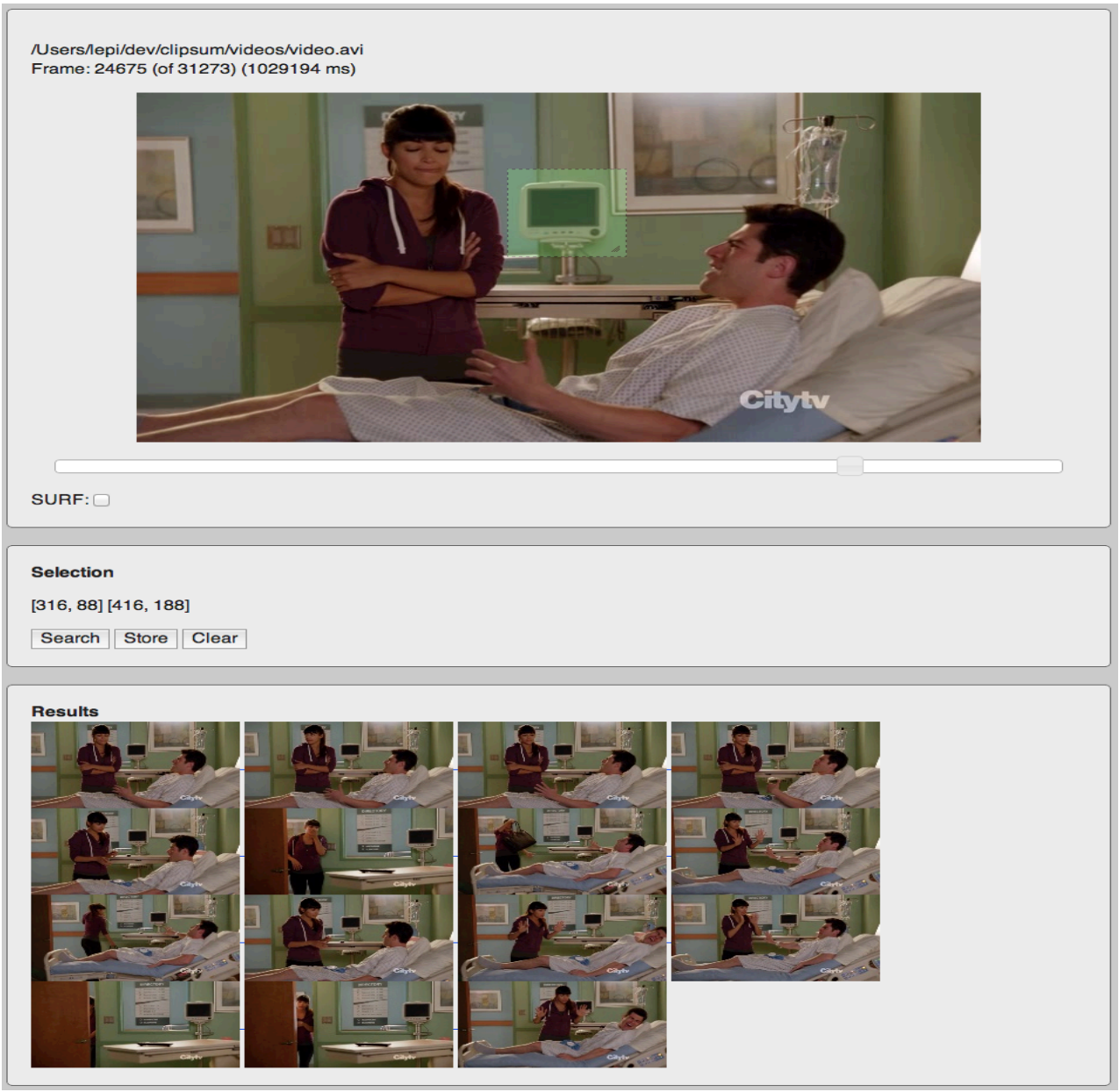

Fig. 5. Snapshot of the implemented system

Without using inverted index the response time in a film with average length the number of all comparisons will be 6000-7000. Since in a comparison the query image and the 700-1000 keypoints of each sample image should have processed, that is why in this case the response time would be more minutes (approximately 5 minutes), which is not acceptable.

In an average film the response time of the implemented scene retrieval is less than a second at case of internal query; at external case the time is approximately a second due to the additional feature extraction process. So the inverted index described in this paper gives a faster solution, the increasing of the speed is more orders of magnitude.

4. Conclusion

In this paper the concept and details of an elaborated video scene retrieval system has been described. Video indexing and retrieval have a wide spectrum of promising applications, motivating the interest of researchers worldwide. There is an overview 8 of the landscape of general strategies in visual content-based video indexing and retrieval.

In our retrieval system the scenes as parts of video are the available set for the end user and for our system as well. The end user first gives a query image via the Graphical User Interface (GUI), and he or she would like to retrieve the most relevant scenes from a long video. Furthermore the user may wonder the results in more details – in less unit than scene, e.g. in sample image – ordered as well. The key problem of such retrieval is the speed of the answer, and the solution for fast search is described in this paper.

An index structure, as contribution of this paper is outlined for speed up. After feature extraction from images an extended clustering algorithm, so called k-means++ has been used for finding similar keypoints in all sample images in the video. The clusters were the base of the inverted index structure, because in this structure each group (cluster) contains the list of the image identities that possess at least 1 keypoint belonging to this cluster. From the query image the keypoints are also extracted, and the corresponding clusters to each query keypoint can be found. The union of these cluster lists is a potential answer set, but this is too large, because many irrelevant images can be found in the hit list as well. The paper shows a solution for filtering the hits, and finally the images will be in descending order based on relevance. At the end of the paper a solution has been shown for ranking of the scenes based on the rank of the images in the hit list by reciprocal rank fusion. Concluding the results it can be stated, that the response time is small due to inverted index structure.

There is a work 13, where users are allowed to submit a short video clip as a query to improve the retrieval reliability. Improvement is achieved by integrating the information about different viewpoints and conditions under which object and scene appearances can be captured across different video frames, so video-based image retrieval (VBIR) will be more reliable than the retrieval using a single image as a query. This idea could be applied in our system as well, so in expansion of the system the user could give the video query, furthermore the interested region in the first and in the last image of the video query.

Acknowledgments

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

References

1. Arthur, A. and Vassilvitskii, S. “k-means++: the advantages of careful seeding”, *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms,* Society for Industrial and Applied Mathematics, pp. 1027–1035, (2007).
2. Bai, Hongliang, Lezi Wang, Yuan Dong, and Kun Tao. "Interactive Video Retrieval Using Combination of Semantic Index and Instance Search." In *Advances in Multimedia Modeling*, pp. 554-556, Springer, Berlin, Heidelberg, (2013).
3. Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. “Speeded-up robust features (SURF). Computer vision and image understanding”, 110(3), pp. 346-359. (2008).
4. Chen, Xu, Alfred O. Hero, and Silvio Savarese. "Multimodal video indexing and retrieval using directed information." *IEEE Transactions on Multimedia,* 14, no. 1, pp. 3-16, (2012)
5. Chergui, Adil, Abdelkrim Bekkhoucha, and Wafae Sabbar. "Video scene segmentation using the shot transition detection by local characterization of the points of interest." In *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on*, pp. 404-411. IEEE, (2012).
6. Elminir, Hamdy K., Mohamed Abu ElSoud, Sahar F. Sabbeh, and Aya Gamal. "Multi feature content based video retrieval using high level semantic concept." *International Journal of Computer Science Issues* (IJCSI) 9, no. 4 (2012).
7. Gao, Wen, Yonghong Tian, Lingyu Duan, Jia Li, and Yuanning Li. "Video Scene Analysis: A Machine Learning Perspective." *In Video Segmentation and Its Applications*, pp. 87-116. Springer New York, (2011).
8. Hu, Weiming, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. "A survey on visual content-based video indexing and retrieval." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 41, no. 6, pp. 797-819, (2011).
9. Jégou, Hervé, Matthijs Douze, and Cordelia Schmid. "Improving bag-of-features for large scale image search." International Journal of Computer Vision 87, no. 3 (2010): 316-336.
10. Lin, Ting-Chu, Jau-Hong Kao, Chin-Te Liu, Chia-Yin Tsai, and Yu-Chiang Frank Wang. "Video instance search for embedded marketing." In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pp. 1-4. IEEE, (2012).
11. Vani, V., and Sabitha Raju. "A detailed survey on query by image content techniques." *Recent Advances in Networking, VLSI and Signal Processing*, CA Bulucea, N. Kalamani, N. Mastorakis et al., eds, pp. 204-209, (2010).
12. Wang, Jianchao, Bing Li, Weiming Hu, and Ou Wu. "Horror video scene recognition via multiple-instance learning." In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 1325-1328. IEEE, (2011).
13. Yang, Linjun, Yang Cai, Alan Hanjalic, Xian-Sheng Hua, and Shipeng Li. "Video-based image retrieval." In *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1001-1004. ACM, (2011).
14. Zha, Zheng-Jun, Meng Wang, Yan-Tao Zheng, Yi Yang, Richang Hong, and Tat-Seng Chua. "Interactive video indexing with statistical active learning." *IEEE Transactions on Multimedia*, 14, no. 1, pp. 17-27. (2012).
15. Zhai, Y., & Shah, M. “Video scene segmentation using markov chain monte carlo.” *IEEE Transactions on Multimedia*, 8(4), pp. 686-697, (2006).
16. Zhai, Yun, and Mubarak Shah. "A general framework for temporal video scene segmentation." In *Tenth IEEE International*

Conference on Computer Vision (ICCV 2005), vol. 2, pp. 1111-1116. IEEE, (2005).
17. Zheng, Yingbin, Renzhong Wei, Hong Lu, and Xiangyang Xue. "Semantic video indexing by fusing explicit and implicit context spaces." In *Proceedings of the international conference on Multimedia*, pp. 967-970, ACM, (2010).
18. Python v2.7.2 documentation, http://docs.python.org/
19. OpenCV, http://docs.opencv.org/
20. Python Imaging Library (PIL), http://www.pythonware.com/products/pil
21. Matplotlib v1.1.0 documentation, http://matplotlib.sourceforge.net/
22. Scientific Computing Tools For Python – Numpy, http://numpy.scipy.org/
23. HDF Group homepage, http://www.hdfgroup.org/HDF5/whatishdf5.html