Automated Content Based Video Retrieval

Srđan Zagorac, Ainhoa Llorente, Suzanne Little,
Haiming Liu, Stefan Rüger

Knowledge Media Institute, The Open University
Walton Hall, Milton Keynes, MK7 6AA, UK
{s.zagorac, a.llorente, s.little, h.liu, s.rueger}@open.ac.uk

Abstract. We describe our experiments for the search task. Eight runs were submitted, all of them corresponding to the fully automated mode, without human interaction in the loop. The system was based on determining the distance from a query image to a pre-indexed collection of images to build a list of results ordered by similarity. We used four different metric measures and two different data normalisation approaches in our runs. We found that the results for all of the runs roughly match the median results achieved in this year's competition.

1 Search Task

The main goal of the search task (SE) is to model a person searching for video segments that contain persons, objects, events, locations, etc. of interest. These elements may be peripheral or accidental to the original subject of the video. The task consists of 24 topics (multimedia statement of information need), and the common shot boundary reference for a search test collection. A ranked list of at most 1,000 common reference shots that best satisfy the need are to be returned from the test collection. The data set used was provided by The Netherlands Institute for Sound and Vision and is a collection of MPEG-1 videos divided into 100 hours of videos for development and 280 hours for test purposes. The collection of videos is divided into shots according to the provided master shot boundary reference and selected keyframe images extracted for each shot. The search engine may work in three different modes, one fully automated (with no human input in the loop), one manually assisted (where one can formulate a one short query) and finally, one interactive (when a human reformulates the query based on topic, query, and/or results). We only deployed fully automated searches.

2 System

Our system is based on determining the distance from a query image to a pre-indexed collection of images to build a list of results ordered by visual similarity [6]. The search engine consists of two logically independent parts, the indexing module and search module, both implemented in JAVA.

The *indexing* module is in charge of index population and manipulation. Each image media object in the index is represented in the form of several colour and texture low-level descriptors (features). Each indexed feature is held in its own sub-index maintained in the memory, the index is permanently stored in Apache Derby DBMS configured as an embedded system.

The indexing module exposes a SOAP based web service endpoint that takes content to be indexed or removed from the index in the form of MPEG7 documents [8] describing each of the images. Each MPEG-7 description contains the source descriptive metadata, such as video and media time point, and the extracted features for the image. During the indexing process all the feature vectors are extracted from the parsed MPEG7 document and added to their respective sub indexes.

The *search* module uses the features to compute the distance between each query object and each indexed object and produces a result list of image references. The search module exposes a SOAP based web service endpoint that takes queries of the query image object in MPQF format. MPQF (MPEG

Query Format) [3] is an XML-based query language developed in the context of the MPEG standards group that defines the format of queries and responses in a distributed multimedia search context. Each MPQF file describes the example or query media object using the same low-level feature set as those in the search engine index.

The search module handles single and multi-image query search requests. A single image query search request contains a feature vector representation of one media object while multi-image query search requests contain several of such media object representation. In case of multi-image query search request the search engine processes obtained result lists through the fusion operator in order to compile the final result list, in contrast to the single image query search request where the result set fusion is not required.

3 Features

The features extracted in all of our submitted runs for this task were a combination of a colour feature, CIELAB, and texture features, Tamura and Gabor.

CIELAB. CIE $L*a*b*$ (CIELAB)[4] emulates human perception and is specified by the International Commission on Illumination (CIE). Its three coordinates represent the lightness of the colour ($L*$), its position between red/magenta and green ($a*$) and its position between yellow and blue ($b*$).

Tamura. The Tamura texture feature is computed using three main texture features called "contrast", "coarseness", and "directionality". Contrast aims to capture the dynamic range of grey levels in an image. Coarseness has a direct relationship to scale and repetition rates and it was considered by Tamura et al. [11] as the most fundamental texture feature and finally, directionality is a global property over a region.

Gabor. Gabor filters are one of the most popular signal processing based approaches for texture feature extraction. These enable filtering in the frequency and spatial domain. A range of filters at different scales and ordinations allows multichannel filtering of an image to extract frequency and orientation information. This is then used to decompose the image into texture features. Our implementation is based on that of Howarth et al. [5].

Due to the fact that Gabor filters can return negative values we make sure during the distance calculation procedure that negative values are excluded from the process.

4 Distances

In our runs we used four different geometric measures to calculate the feature based distance between the query media object (q) and the indexed media object (m) - Euclidean, Manhattan, Canberra and Squared chord distance.

Euclidean and Manhattan (d_p) [6]

These metric measures belong to *Minkowski family* of distances.

$$d_p(A,B) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}} \tag{1}$$

The Minkowski distance is a general form of the Euclidean (p=2), Manhattan (p=1) and Chebyshev ($p = \infty$) distances. Here $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$ are the query vector and test object vector respectively.

Canberra Metric (d_{can}) and Squared Chord (d_{sc}) [9]

The *Canberra distance*

$$d_{\mathrm{Can}}(v,w) = \sum_{i=1}^{n} \frac{|v_i - w_i|}{|v_i| + |w_i|} \tag{2}$$

is very sensitive for components near zero. Note that the term $|v_i - w_i|/(|v_i| + |w_i|)$ needs to be replaced by zero if both v_i and w_i are zero.

A less usual measure is given by the *squared chord dissimilarity*,

$$d_{\mathrm{sc}}(v,w) = \sum_{i=1}^{n} (\sqrt{v_i} - \sqrt{w_i})^2, \tag{3}$$

which seems to have been used in paleontological studies and in pollen data analysis, both with little theoretical justification. However, in comparative evaluation the squared chord measure does remarkably well [6]. Please note that the squared chord dissimilarity cannot work with negative components; it should be replaced with a modified version

$$d_{\mathrm{msc}}(v,w) = \sum_{i=1}^{n} \left(\mathrm{sign}(v_i)\sqrt{|v_i|} - \mathrm{sign}(w_i)\sqrt{|w_i|}\right)^2 \tag{4}$$

instead.

In the search time each one of three indexed image features is taken and the distance against the query's equivalent feature vector is computed. The final distance between an indexed image and the query image is derived by totaling all three feature distances.

$$d_{ft} = \sum_{f=1}^{f} d_f, \tag{5}$$

5 Normalisation

In our search runs we applied two normalisation approaches to the distances and/or query objects.

Approach 1. During the search index creation we select a random subset (X) to be used in normalising the distances. For each indexed feature a distance between the query vector and the index features is calculated and the following formula for normalisation is applied:

$$\hat{d_{qm}} = \frac{d_{qm} - \mu_X}{\sigma_X}, \tag{6}$$

where mean (μ_X) and standard deviation (σ_X) are calculated from the pairwise distances of the random sample of the vectors (X) from the index.

Approach 2. Each vector in the index and each query vector is normalised using the following formula:

$$v_i^{'} = \frac{|v_i|}{\sum_{i=1}^{n} |v_i|} \tag{7}$$

6 Result Set Fusion

Submitting each query media object to the search engine produces a ranked list of images based on the normalised distance measure. When a query consists of a set of media objects a technique for aggregating or fusing these results lists is needed. To achieve this we have implemented the Borda algorithm [1] which treats each query media object as a "voter" who submits a set of c candidates in order of preference. In our implementation each keyframe ID consist of shot ID (p) and keyframe ID (c). For example p001:c012 denotes shot 001 and its keyframe 012. Our fusion procedure consists of two loops. In the first loop we run each result list through the fusion operator to create a concise list without repeating shot IDs. We also recreate the IDs by keeping the actual shot ID (p) and removing the key frame ID (c). In this way the result list becames the list of shots and not keyframes anymore. In the second loop the same algorithm is applied to fuse all the result lists into one final result list.

Rank	**RS 1**	**Fused RS**	**Fused Rank**
1	p001:c008	p001	1
2	p002:c011	p002	2
3	p001:c002	p006	3
4	p006:c025	p0023	4
5	p006:c022		
6	p023:c002		
7	p0023:c012		

Table 1. First loop - result set fusion

Table 1 shows a sample first loop of the Borda fusion procedure where one result list with multiple shot occurances is fused and accordingly re-ranked into a list of shot IDs.

Rank	**RS 1**	**RS 2**	**RS 3**	**Fused RS**	**Fused Rank**
1	p001	p002	p023	p002	1
2	p002	p012	p002	p001	2
3	p006	p005	p007	p023	3
4	p023	p001	p012	p005	4
				p006	5
				p007	6
				p012	7

Table 2. Second loop - result set fusion

Table 2 shows a sample second loop of Borda fusion procedure where three result lists are fused and accordingly re-ranked.

7 Experiments & Results

We submitted eight runs to TRECVID in order of expected performance:
MMIS2SQ - used Squared Chord distance with normalisation approach 1,
MMIS2CAN - used Canberra distance with normalisation approach 1,
MMIS2MAN - used Manhattan distance with normalisation approach 1,

MMIS2EUC - used Euclidean distance with normalisation approach 1,
MMIS2A-SQ - used Squared Chord distance with normalisation approach 2,
MMIS2A-CAN - used Canberra distance with normalisation approach 2,
MMIS2A-MAN - used Manhattan distance with normalisation approach 2,
MMIS2A-EUC - used Euclidean distance with normalisation approach 2.

Each run was completely automated using a content-based query by example in multi query scenario as described in Section 2 with no human input to the search process. A simple SOAP web service client was used to submit queries to the search engine and collect the fused result set. The whole search process was performed as a consecutive sequence without any adjustments to the queries or the result sets.

The process is described as follows. First, the data set was prepared to populate the search engine index with the feature set extracted from the key frames of the test data set. Each test media example was processed to extract a frame from the beginning, middle and end of each shot contained within the media example. Shot boundaries were taken from the TRECVID master shot boundary document. Each extracted frame contained the shot id information in its file name.

Each key frame was analysed using our feature extraction tool to generate the required features. Using a tool developed for the PHAROS project [2], the output from these two processes was converted into an MPEG-7 file describing the key frame including the timestamp and the three chosen features. Each MPEG-7 file was then loaded into the search engine index resulting in an index of approximately 280,000 files.

Second, the query descriptions were processed to generate the MPQF files as described in Section 2. The video media examples for each query were processed to extract a frame from the beginning, middle and end of the example to improve the query coverage. The resulting key frames for each video and the image media examples were processed to extract the required features. A set of MPQF files was produced for each of the 24 query topics.

Each query was run in sequence through the system described in Section 2.

Once the search engine process had finished, it produced a text file containing an ordered list of shot references from the search index generated from the fused ranked list for each example media. This output file was then processed to output XML document as prescribed by submission DTD.

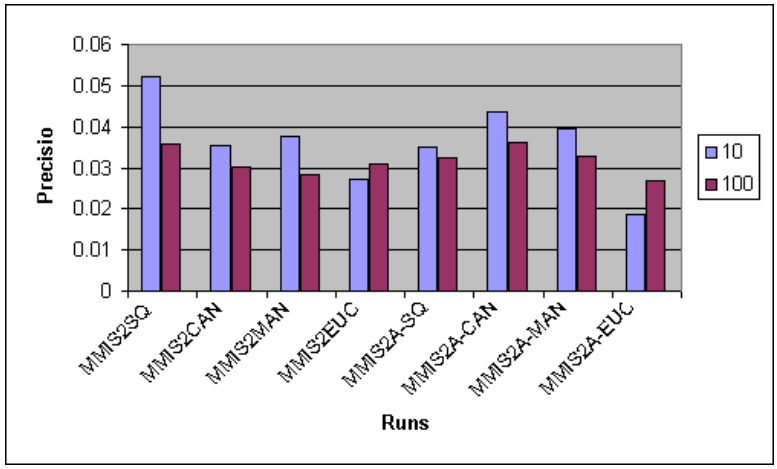

Fig. 1. Precision at 10 and 100 shots for all runs

Figure 1 shows average precision at 10 and 100 shots for all of our submitted runs. It is evident that there is no clear winner and the all of the runs remain in 0.03 to 0.04 average precision range. After performing a statistical significance test (a non-parametric Friedman test as proposed by Hull [7]) on our eight proposed methods compared on 24 queries and with a significance level of 5%, we conclude that the

differences between our submitted runs are not statistically significant. This makes sense as all of them correspond to variations of the same algorithm.

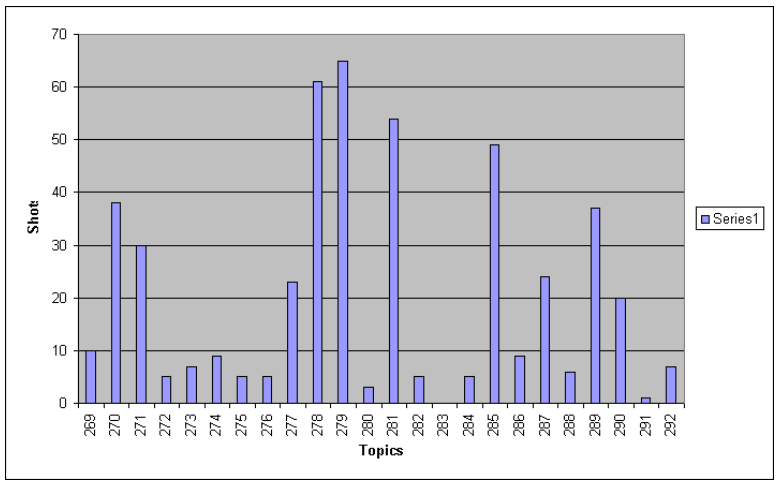

Fig. 2. Precision at 100 shots for run MMIS2SQ

In Figure 2, we show the performance of our first run (MMIS2SQ). In general it performs the best in topics that depicts people in various situations such as topic 0279 "Find shots of people shaking hands", topic 0281 "Find shots of two more people, each singing and/or playing a musical instrument", topic 0289 "Find shots of one or more people, each sitting in a chair, talking" and rich outdoor scenes such as topic 0270 "Find shots of a crowd of people, outdoors, filling more than half of the frame area", topic 0271 "Find shots with a view of one or more tall buildings (more than four stories) and the top story visible " and topic 0278 "Find shots of a building entrance".

In contrast topics like topic 0291 "Find shots of a train in motion, seen from outside", topic 0280 "Find shots of a microscope " or topic 0276 "Find shots of one or more dogs, walking, running, or jumping" achieved the worst results.

8 Conclusions

Our results for all of the eight runs roughly match the median results achieved in this year's competition. On the whole they are within our expectations and have made clear to us that this combination of features has reached its limit and we need to explore new features used in search.

For the next TRECVID competition we plan to develop a new shot boundary algorithm that will further streamline and improve precision of our key frame extraction procedure. In addition to that, we will reduce our search time by implementing one of the fast approximate nearest neighbors algorithms such as k-means algorithm and kd-trees algorithm [10]. We will also put emphasis on improving our existing content based search interface that has not been used in this year's competition and with the aim of participating in the interactive search task next year.

Acknowledgements: This work was partially funded by the PHAROS project sponsored by the Commission of the European Communities as part of the Information Society Technologies programme under grant number IST-FP6-45035 and by Santander Universities.

References

1. J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of International ACM SIGIR Conference on Research and Development in Informational Retrieval*, 2003.
2. S. Debald, W. Nejdl, F. Nucci, R. Paiu, and M. Plu. Pharos platform for search of audiovisual resources across online spaces. In *SAMT2006*, 2006.
3. M. Gruhne, P. Dunker, M. Döller, and R. Tous. Distributed cross-modal search within the mpeg query format. *Image Analysis for Multimedia Interactive Services, International Workshop on*, 0:211–214, 2008.
4. A. Hanbury and J. Serra. Mathematical morphology in the CIELAB space. *Image Analysis & Stereology*, 21:201–206, 2002.
5. P. Howarth and S. Rüger. Evaluation of texture features for content-based image retrieval. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 326–324. Springer-Verlag, July 2004.
6. R. Hu, S. Rüger, D. Song, H. Liu, and Z. Huang. Dissimilarity measures for content-based image retrieval. In *Proceedings of IEEE International Conference on Multimedia and Expo*, June 2008.
7. D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338, New York, NY, USA, 1993. ACM.
8. H.-G. Kim, N. Moreau, and T. Sikora. *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval.* John Wiley & Sons, 2005.
9. M. Kokare, B. Chatterji, and P. Biswas. Comparison of similarity metrics for texture image retrieval. In *TENCON Conference on Convergent Technologies for Asia-Pacific Region*, pages 571–575, Vol. 2, Oct. 2003.
10. M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.
11. H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Trans on Systems, Man and Cybernetics*, 8(6):460–472, 1978.