Brno University of Technology at TRECVid 2009

Petr Chmelař, Vítězslav Beran, Adam Herout,
Michal Hradiš, Ivo Řezníček, Pavel Zemčík

Brno University of Technology
Faculty of Information Technology
Božetěchova 2
Brno, 612 66
Czech Republic

Abstract

In this paper we describe our experiments in High-level feature extraction (HLF) and Search tasks of the 2009 TRECVid evaluation. This year, we have concentrated mainly on the local (affine covariant) image features and their transformation into a searchable form, especially using the indexing techniques.

In brief, we have submitted the following runs:

1. HLF: We have used training method based on support vector machine (SVM) using five types of global and local image features. Results were submitted in the BRNO_HLF_SI run.
2. Search: We have performed a fully automatic experiment based on the transformed local image features together with face detection and global features – color layout and texture features in the BrnoUT_visual.2 run.

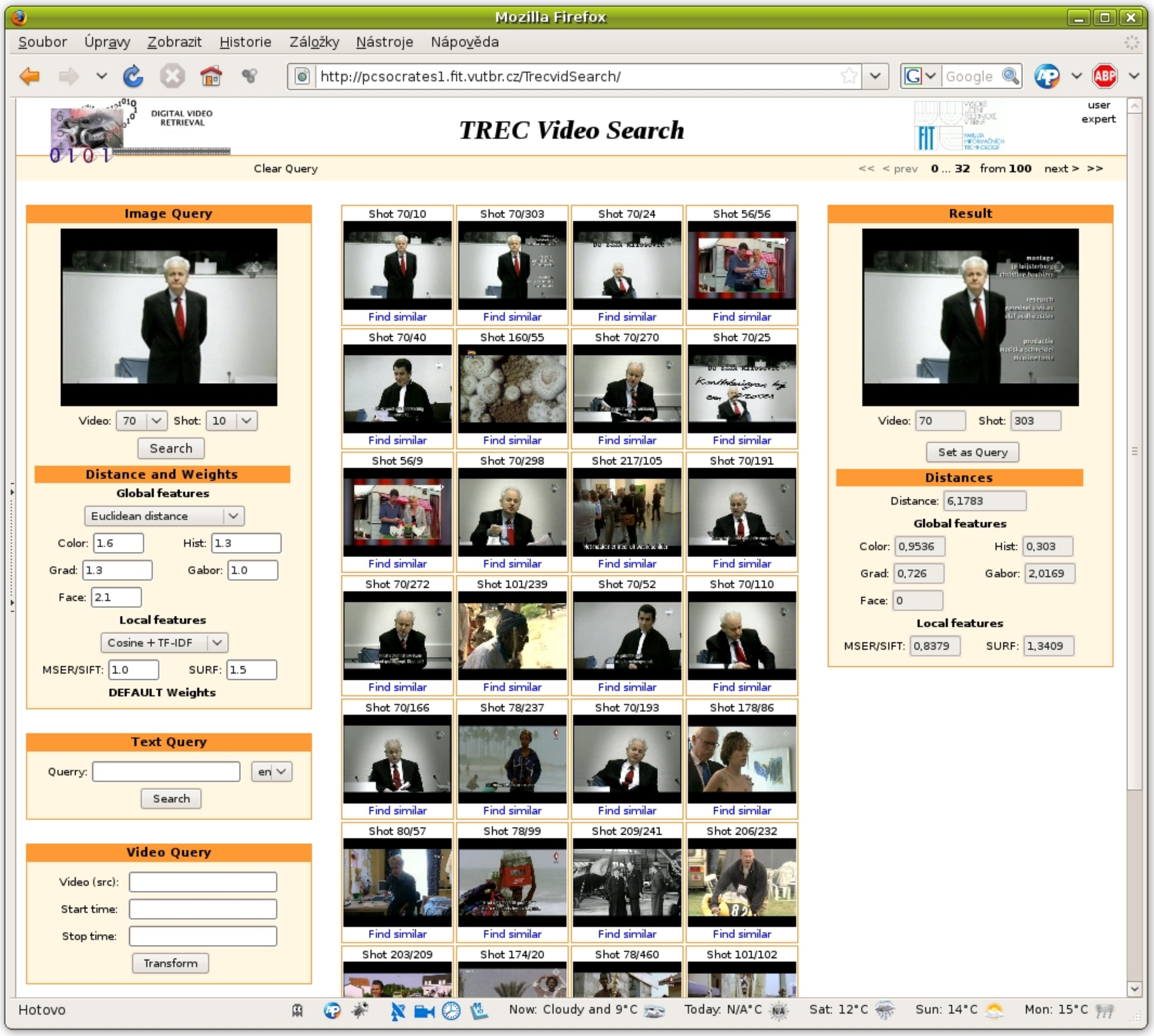
The paper is organized as follows. In Section 1, a motivation and an overview of the work is presented. We dedicated Section 2 to the feature extraction task, which is being used in common by the HLF and Search tasks. Details of the tasks we have sent are given in Section 3 and Section 4. Finally, Section 5 discusses the achieved results and concludes the paper. The result tables are attached at the end of the paper.

Acknowledgements

This work has been supported by the project EU-6FP-IST, IST-033812-AMIDA, by the research project "Security-Oriented Research in Information Technology" CEZMSMT, MSM0021630528, and “Centre of Computer Graphics” (CPG-LC06008), Czech Ministry of Education, Youth, and Sports grant. This project has been performed also with a financial support from the Czech Republic state budget through the Ministry of Industry and Trade.

1. Introduction

Brno University of Technology, Faculty of Information Technology has taken part at TRECVid third time in a row this year [1]. In the past three years, we have developed a lot of software, specifically many shell and Perl scripts, programs written in C for video and feature processing as well as PostgreSQL native functions and a Java program for video retrieval which has a simple interface and a web demonstration application (see Figure 1 below).



http://pcsocrates1.fit.vutbr.cz/TrecvidSearch/

Image Query: Video: 70 Shot: 10

Global features: Euclidean distance

Color: 1.6 Hist: 1.3
Grad: 1.3 Gabor: 1.0
Face: 2.1

Local features: Cosine + TF-IDF

MSER/SIFT: 1.0 SURF: 1.5

Text Query: en

Result: Video: 70 Shot: 303

Distance: 6,1783

Global features
Color: 0,9536 Hist: 0,303
Grad: 0,726 Gabor: 2,0169
Face: 0

Local features
MSER/SIFT: 0,8379 SURF: 1,3409

Figure 1. A screenshot of TRECVid Search web demonstration application in expert mode.
See also http://www.fit.vutbr.cz/research/prod/index.php.en?id=73.

2. Sound and Vision dataset preprocessing

The feature extraction for High-level and Search tasks have a lot in common; thus they are presented along. Before the particular tasks are described, the low-level feature extraction, face detection, clustering, and indexing search techniques used are described.

Feature extraction

Feature extraction is an automated process of extracting structured information from the unstructured data, both representing the same object. Its output is a feature vector characteristic for the data (and objects contained in the data). In our case, the data is a multimedia shot – a sequence of frames between two cuts. The shot may be represented through several keyframes, camera and object motion, or through speech and sound. This paper is focused only on visual features. They can be either global – concerning the whole video frame (e.g. color histograms, layout, or an amount of motion), or local – concerning regions smaller than the frame. Video search based on local feature descriptors currently seems to outperform other approaches [1].

Color histogram Based on HSV color model

The color histogram contains statistical information about color distribution in terms of frequency of hues and saturations in the frame (using HSV color model). The better spatial description is achieved by dividing the frame into several patches. The frame division is not adaptive, so the patches have similar size. Each patch is processed separately; the histogram is computed and normalized.

Multi-scale gradient distribution

The histogram of gradient orientations serves as other part of the feature vector. First, the frame gradients are computed. Then, each gradient contributes to the histogram bin according to its orientation. The contributions are weighted using the gradient magnitude. The gradients are computed on different frame resolutions so also lower frequency structures contribute to final feature vector. The resolution of both color histogram and gradient histogram, the resolution of the frame grid, and the frame scale levels are the descriptor parameters.

Color layout

The color layout computation is based on JPEG compression technique. First, the image is resampled into 8x8 pixels in Y'CbCr color model. Then, the discrete cosine transform (DCT) is applied on each channel. The descriptor coefficients are then extracted in zig-zag manner [12] as illustrated in Figure 2. We use 20 (Y) + 15 (Cb and Cr) coefficients; thus the feature vector has 50 coefficients.

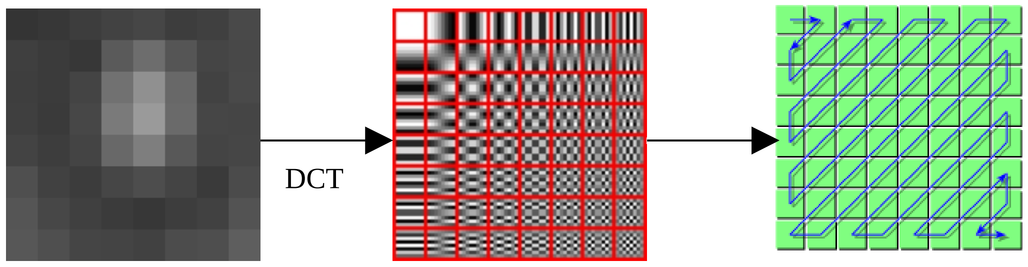
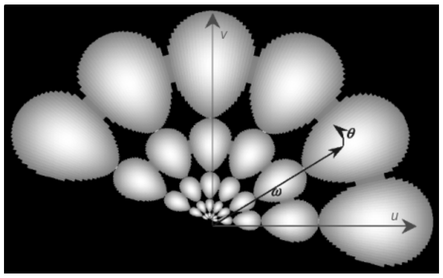
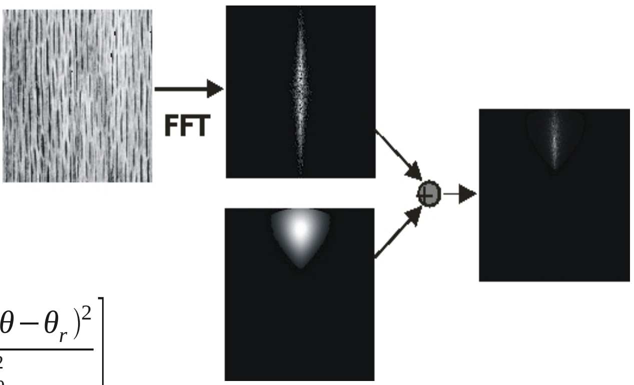


Figure 2. Illustration of the color layout description process.

Gabor texture

Using a bank of Gabor filters [13] in the frequency domain, we can divide the space created eg. using Fourier transform, into bands, as illustrated in Figure 2 above. We use the first moments of energy in the filtered 30 sub-bands (G_P) – 6 angular (*30°*) and 5 radial (in octaves) for construction of the descriptor.





$$G_{P_{s,r}}(\omega,\theta) = \exp\left[\frac{-(\omega-\omega_s)^2}{2\sigma_{\omega_s}^2}\right]\exp\left[\frac{-(\theta-\theta_r)^2}{2\sigma_{\theta_r}^2}\right]$$

Figure 3. Illustration of the Gabor texture description process.

Local features

Local features represent visual objects as a compound structure of statistically interesting regions, such as points, edges, or homogenous regions (in color) [14]. The local feature extraction process consists of two steps – detection of remarkable objects in images and their description. The main property of these steps is the repeatability – an ability to detect and describe the same object uniformly under various fotometric conditions, geometric conditions, and noise. We have employed two types of detectors and descriptors.

Maximally Stable Extremal Regions

Maximally Stable Extremal Regions (MSER) have been used for finding of connected components of an appropriately thresholded image (experimentally) to be maximally stable. Extremal in this context means that all the pixels inside the region have lower intensity (are darker) or higher intensity than pixels at the edge of the region [14].

Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) [14] has been used for description of regions found by MSER. It captures an information (about an elliptic neighborhood) of the point of interest (center of the region) using a histogram of locally oriented gradients and stores it as a vector of size 128 (8 orientations, each in 4x4 locations).

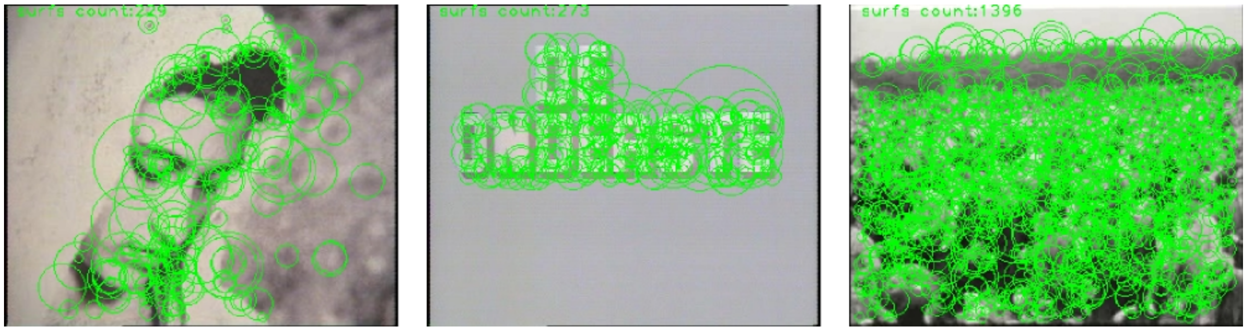


Figure 4. An Illustration of affine covariant regions (SURFs) on different high level concepts – face, text and crowd.

Speeded Up Robust Features

Speeded Up Robust Features (SURF) is used for the detection of interesting points based on determinant of Hessian matrix (second partial derivatives) of an integral image. The description is based on Haar transformation, similarly to the SIFT descriptor. It is illustrated in Figure 4.

Frame signatures

Visual Vocabulary was used for vector quantization of local feature descriptors. The vocabularies were constructed separately for SIFTs and SURFs in training stage. We used kd-tree for data search and k-means to detect clusters. The sizes of vocabularies were 1k for TRECVID purposes. The video frames were then described as bag-of-words that is unordered set of weighted visual words. The used weighting scheme was standard tf-idf [22].

Face detection

The knowledge about presence of people in a video is a valuable source of information for both Search and High-level feature extraction tasks. Many possible ways of detecting people in image exist, one of which is the face detection. The face detection task has been well studied and many methods for reliable real-time detection exist. The most successful are appearance-based methods which use some variation of a cascade of simple boosted classifiers to scan the images. This approach was originally proposed by Viola and Jones in 2001 [20].

We have used a frontal face detector to extract four low-level features from the video frames. One of the features was the total number of faces present in a video frame. The other three features were the numbers of small, medium, and large faces in the frame. The rationale behind this choice was that the number and sizes of faces are more informative then their positions and that the number of detections is too small to make this information even sparser by considering the position.

Four classifiers in total with random initial choice of training samples were created by the WaldBoost algorithm [21]. During detection, the responses of the individual classifiers were integrated thriough weighted voting to get the final detections. Instead of the traditional Haar-like features[20], the Local Rank Patterns (LRP) [4] were used as features by the classifiers. The LRP features were chosen because they show better performance on the frontal face detection task [4] than the Haar-like features especially in combination with the slower and more precise classifiers which were used for this particular task. The classifiers were created using our experimental framework for research on detection classifiers [5].

3. High-level feature extraction

The most important parts of the system in the figure below are these:

a) Low-level feature extractors – These are described in section 3.1. Several feature extractors are employed; their feature vectors are concatenated to make a per-frame feature vector. Note, that the features are relatively generic. Feature range normalization is part of the feature extraction process.
b) SVM training and cross-validation – Using grid-search, the SVM kernels are optimized and for each high-level feature to be searched, one model is selected by the Model Selection module.
c) Per-frame SVM evaluation – Evaluates the low-level feature vector for each frame in the testing dataset based on the selected SVM model for each high level feature.
d) Per-shot decision – Judges the set of per-frame classifications based on the shot-boundary reference to make a decision on each shot of the testing video dataset.

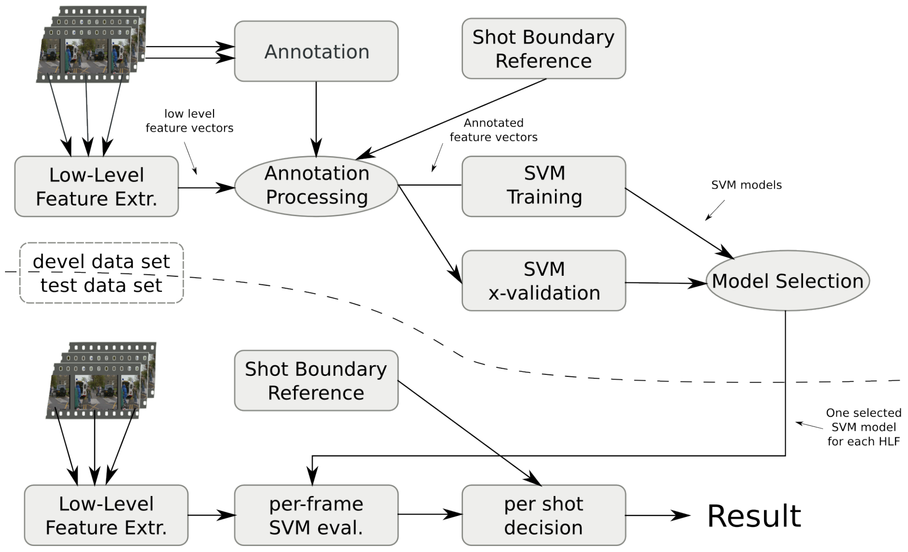


Figure 5. The structure of the high-level feature extraction system.

The system can be described as a brute-force approach to high-level feature extraction since the low-level features are rather generic than built to match specially the high-level concepts looked for. Also, the main classification machine (SVM) is generic. The only specialized part of the system is the per-shot decision making subsystem which constitutes a very simplistic decision tree.

Composed per-frame feature vector and its processing

All the partial feature vectors mentioned in the previous text are concatenated to make a per-frame feature vector which serves as input to the per-frame classifier described in the next paragraph. This feature vector is normalized across the whole data-set, i.e. maxima and minima for separate features are found in the training dataset and the features are independently rescaled so that these maxima and minima correspond to 0.0 or 1.0 respectively. The scaling factors are used for the test data, which can then (scaled) exceed the normal interval (0-1). These infrequent cases were not found harmful for evaluation using the SVM classifier. These operations were significantly sped up by using database storage where all features, all videos metadata, and embedded rescaling functions are stored.

Per-frame SVM classifier

LIBSVM [10] was used for classification of separate frames – one classifier was trained for each high level feature to be detected. Given that, shot annotations were used for all the frames in a given shot. The SVM classifier was trained on 70% of the training data provided and cross-evaluated on the resting 30%. This cross-evaluation was used for selection of proper parameters of the SVM kernels. The SVM training and parameters selection took majority of the development time (thousands of hours sequentially) and this would be the part most likely to get speeded up in future versions of the system.

Per-shot decision based on per-frame results

For each high level feature separately, the results from per-frame classification were judged for each shot (dropping frames at the beginning and end of each shot to avoid misclassification). Two quantities were observed in this decision process:

- Positive rate $R = N/P$, where P is the number of positively judged frames and N is number of all frames (excluding the dropped initial and tailing frames in each shot).
- Largest positive sequence s is the length of the longest sequence of positively judged frames.

Two thresholds are defined for these quantities rt, st; exceeding either of them selected the shot being judged for output. These two threshold values were found experimentally on the testing data. The output (positively classified) shots were sorted by the value of r; in cases their number would have exceeded the given TRECVID limit of positive shots, shots with largest r were selected.

Results, future work

The results of our system for the high-level feature extraction task in TRECVid 2009 were below average. We would have expected significantly better results, if we had the time to employ our object classifiers trained for different classes of objects then faces (vehicles, planes, animals), text and local features entering the ensemble classification.

4. Search

The video retrieval is based on similarity search in a database containing the video metadata. A video query has a form of an example – a multimedia object (text, image video) from which visual or high-level conceptual features are extracted. The features are then compared to the features in the database that represent objects in video. The objects represented in the database by the most similar features are selected as a result of the query.

The scheme of the developed system is illustrated in the figure below. It includes the user (NIST) of the system who provides data and the queries and is a consumer of the results. The video data is being (automatically) annotated in the same manner as in the High-level feature extraction task. So the database is filled with per keyframe feature vectors including color and texture descriptors, faces found in the image as described in section 3.1. and other data provided by NIST.

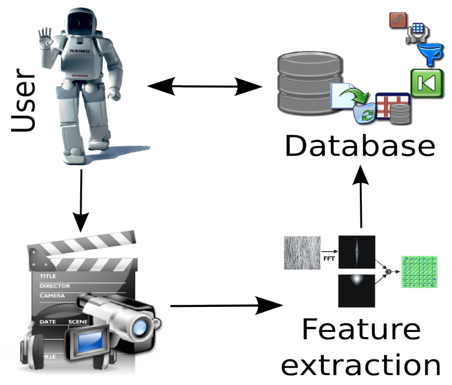


Figure 6. The structure of the multimedia retrieval system.

Techniques

For the content-based copy detection and search tasks, before all, we have used the PostgreSQL database system. There we store all extracted features, video and shot metadata, annotations, ASR and MT data.

For the fixed-length (low-level) features, we have employed standard Euclidean distance (*p=2* in) $eucd(d_1, d_2) = \left(\sum_{i=1}^{n} \left| d_1[i] - d_2[i] \right|^p \right)^{\frac{1}{p}}$. For the variable-length, ones the cosine distance $cosd(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1||d_2|}$. In the cosine distance the (words) weighting is performed using TF-IDF: *tf-*

idf(w) = tf(w)idf(w), where $tf(w) = \frac{|d(w)|}{|d|}$, $idf(w) = \log\left(\frac{|D|}{|D(w)|}\right)$. We also use the Generalized Inverted (document) Index (GIN, [16]) to speed up the queries. This is illustrated in the figure below.

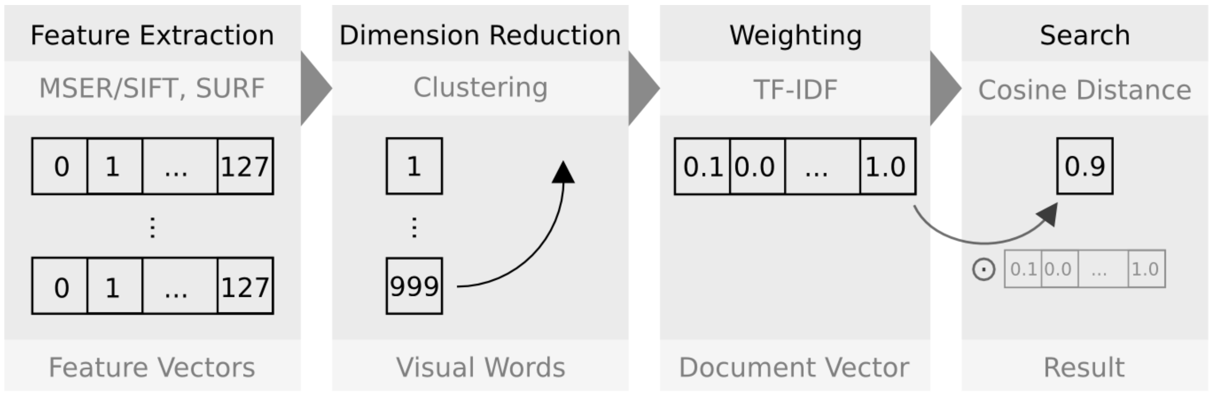


Figure 7. The visual document retrieval process.

However, these techniques were unable to find anything in case of the local features in serious time; thus we had to employ some reduction of the search space, similarly to the [18]. However, in that approach the dictionary construction (clustering) is even much more severe, because the search can be parallelized. Thus we have used another one described next.

Voronoi tessellation based clustering of the local features

Although, many clustering techniques exist, it is not possible to use them for all purposes. The initiatial task was to create as many clusters as possible (eg. thousands) for the local image features description in huge amount of video for Content-based copy detection and Search tasks. We have obtained 25 mil. of MSER/SIFT [14] feature vectors (32 GB) and 38 milion SURF [15] descriptors (41 GB) of local invariant features, as described in section 3.1. These large dimensional vectors cover the space almost continuously and commonly used clustering methods are unable to create enough classes or to finish in serious time. For that purpose we have used (also modified) versions of k-Means and DBSCAN clustering methods [8]. However, in the related literature, the problem is mentioned to have solution only for approximately 1GB of the data, when thousands of classes are needed.

Therefore, we have designed a new method based on Voronoi tessellation [7] that needs no more than two passes through the data. The approach is based on discovery of clusters in higher density locations. Because of the large dataset, it is possible to create higher amount of candidate clusters and select appropriate number of classes (large but not huge) and the rest data assign to these classes. The method has been implemented as a set of SQL functions and queries, the pseudo code follows:

```
Algoritm 1: Candidate classes discovery.

      for each ( SELECT f.vector_id, f.vector
                 FROM fvectors as f ) {
         SELECT c.cluster_id,
                distance( f.vector, c.vector ) AS dist
         FROM clusters AS c
         ORDER BY dist LIMIT 1;
         if ( dist > mindistance )
            INSERT INTO clusters
            VALUES ( next(cluster_id), f.vector, 1 );
         else
            UPDATE clusters SET n_vectors += 1
            WHERE cluster_id = c.cluster_id
      };
```

Algorithm 2: Two variants of class selection (third is using a clustering method :)

```
      DELETE FROM clusters
      WHERE n_vectors < minvectors
         OR
         cluster_id NOT IN (
            SELECT cluster_id FROM clusters
     ORDER BY n_vectors DESC LIMIT maxclusters );
```

Algorithm 3: Clusters assignment.

```
      for each ( SELECT f.vector_id, f.vector
                 FROM fvectors as f ) {
         SELECT c.cluster_id,
                distance(f.vector, c.vector) AS dist
         FROM clusters AS c
         ORDER BY dist LIMIT 1;
         UPDATE fvectors SET cluster=c.cluster_id
         WHERE vector_id = f.vector_id;
      };
```

The approach has been tested on a huge problem and a large amount of classes. Performed experiments (to be published) have proven that the new approach is significantly faster than the traditional techniques (linear complexity). The tf-idf weighting and cosine distance have been then used to accomplish the task.

5. Achieved results and conclusions

High-level feature extraction

Our solution can be generally described as a brute-force approach which relies on generic software pieces (several feature extractors, SVM library, training/evaluation framework for distributed computing), that solve the task in an "uninformed" way. We have sent one result based on the global low-level features, the face detector and bag-of-word like sparse features.

For future implementations of HLF extraction for TRECVID or similar evaluations, we intend to include more object detectors, other types of features, and similar frame-processing engines to provide specialized and "informed" knowledge to the overall classification process. These will be represented as mid-level features entering the per-frame classifier. Also, many speed-up optimizations have been suggested from the undertaken runs, which would enable more experimenting for future implementations.

Search

The BrnoUT_visual.2 run is based only on global (color, texture) and local (MSER/SIFT, SURF) image features and aggregated face descriptor. The run performance was average, which is surprising after good results last year. However, this was caused by the fact that we have not innovated the techniques much since the last year because of the lack of the manpower and the TRECVid community has moved forward very much.

Overall conclusions

We have to thank all the people in NIST and groups providing data, transformed data, video and shot references, speech, translations, keyframes, annotations, evaluation metrics, and all the human and computer power. We think that this is the real force of TRECVid, together with the inspiration from and of all the participants and groups.

References

[1] Smeaton, A. F., Over, P., and Kraaij, W. 2006. Evaluation campaigns and TRECVid. In Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval (Santa Barbara, California, USA, October 26 - 27, 2006). MIR '06. ACM Press, New York, NY, 321-330. DOI = http://doi.acm.org/10.1145/1178677.1178722.

[2] Beran, V. Hradiš M. Zemčík P. Herout A and Řezníček I. 2008. Video Summarization at Brno University of Technology. In Proceedings of the TRECVID BBC Rushes Summarization Workshop 2008.

[3] Chmelař P., Hernych R., Kubíček D. 2008. Interactive Visualization of Data-Oriented XML Documents, In: Advances and Innovations in Systems, Computing Sciences and Software Engineering, IEEE, Springer.

[4] Hradis, M., Herout, A., Zemcik, P. 2008. Local Rank Patterns - Novel Features for Rapid Object Detection, In: Proceedings of ICCVG 2008.

[5] Hradis M. 2008. Framework for Research on Detection Classifiers. In: Proceedings of Spring Conference on Computer Graphics, Budmerice, SK, 2008, s. 171-177.

[6] Mlích, J. And Chmelař, P. 2008. Trajectory classification based on Hidden Markov Models, In: Proceedings of 18th International Conference on Computer Graphics and Vision, Moscow. p. 101-105, ISBN 595560112-0.

[7] Aurenhammer F. – Klein R. 2000. Voronoi Diagrams. Handbook of Computational Geometry, 1 ed. North Holland. ISBN 0444825371.

[8] Berkhin P. 2006. A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data. p 25-71, ISBN 978-3-540-28348-5.

[9] Carmona E.J., Martinez-Cantos J. and Mira J. 2008. A new video segmentation method of moving objects based on blob-level knowledge. Pattern Recognition LettersVolume 29. p 272-285.

[10] Chang C.C. and Lin C.J. 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] Fellbaum, C. 1998. WordNet: An Electronic Lexical Database. MIT Press. ISBN 978-0-262-06197-1.

[12] International Telecommunication Union. 1992. Information Technology – Digital Compression and Coding of Continuous-tone Still Images – Requirements and Guidelines. http://www.w3.org/Graphics/JPEG/itu-t81.pdf.

[13] Ho Kyung et al. 2001. MPEG-7 Homogeneous Texture Descriptor. ETRI Journal 23: 41-51. DOI = 10.1.1.17.7368.

[14] Mikolajczyk K. et al. 2005. A Comparison of Affine Region Detectors. International Journal of Computer Vision 65, no. 1 p. 43-72.

[15] 1.Bay H. – Tuytelaars T. – Van Gool L. 2006. SURF: Speeded Up Robust Features. Computer Vision – ECCV 2006, p 404-417.

[16] PostgreSQL Global Development Group. 2008. PostgreSQL 8.3 Documentation: GIN Indexes. http://www.postgresql.org/docs/8.3/static/gin.html.

[17] Van Rijsbergen C. J. 1979. Information Retrieval. Butterworth-Heinemann. ISBN 0408709294. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[18] Sivic J. – Zisserman A. 2008. Efficient Visual Search for Objects in Videos. Proceedings of the IEEE 96, no. 4. ISSN 0018-9219.

[19] Intel Corporation. 2005. The Open Computer Vision Library. http://opencvlibrary.sourceforge.net/.

[20] Viola, P., Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. In: CVPR.

[21] Sochman, J., Matas, J. 2005. WaldBoost - Learning for Time Constrained Sequential Detection. In: CVPR, (2), 2005, s. 150-156.

[22] Shih-Fu Chang et al., High-Level Feature Extraction and Interactive Video Search. http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/columbia.pdf

TRECVID 2009: Feature extraction results

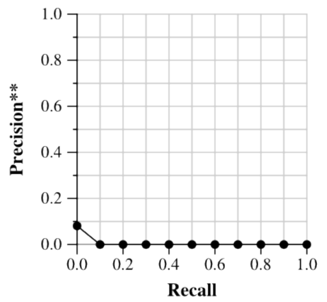
Run ID:	A_BRNO_HLF_SI_1
Processing type:	Automatic
System training type:	A (common devel.data/annotation (+))
Priority:	1

PLEASE NOTE: ALL OF THE MEASURES BELOW ARE BASED ON ASSESSMENT OF A 50% RANDOM SAMPLE OF THE NORMAL SUBMISSION POOLS

Across 20 test features

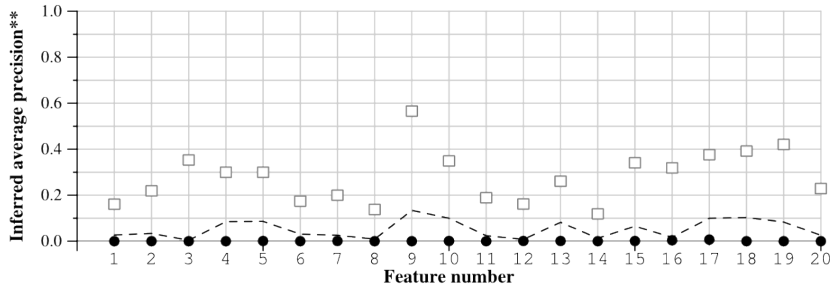
Total true shots*: 7036
Total true shots returned*: 142

Mean(inferred average precision)**: 0.001



Interpolated recall	precision**
0.0	0.0810
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision** at n shots	
5	0.0000
10	0.0200
15	0.0200
20	0.0300
30	0.0400
100	0.0240
200	0.0150
500	0.0104
1000	0.0088
2000	0.0070



Run score (dot) versus median (---) versus best (box) by feature

* actual counts from a 50% random sample of the normal submission pools
** estimate using 50% sample (e.g., estimated precision = 2 * actual from sample) estimated precision may exceed 1.0

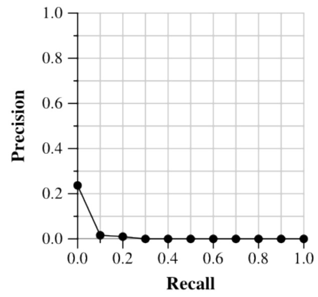
TRECVID 2009: search results

Run ID:	BrnoUT_visual
Processing type:	automatic
System training type:	C (other than type A)
Condition:	N (normal search)
Priority:	2

Across 24 test topics (269-292)

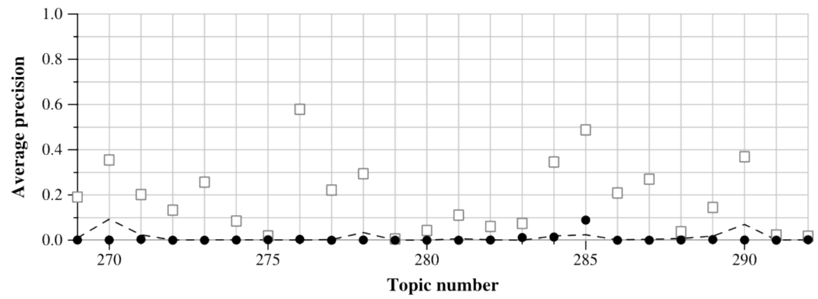
Total relevant shots: 10619
Total relevant shots returned: 501

Mean(prec. @ total relevant shots): 0.024
Mean(average precision): 0.006

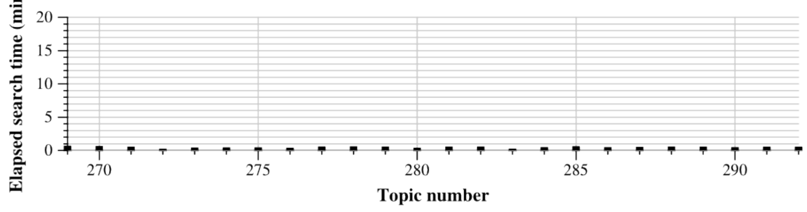


Interpolated recall	precision
0.0	0.2370
0.1	0.0162
0.2	0.0101
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.1083
10	0.0792
15	0.0722
20	0.0708
30	0.0597
100	0.0429
200	0.0346
500	0.0261
1000	0.0209



Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic