TNO instance search at TRECVID 2010

John Schavemaker, Peter Jan Doets, Stephan Raaijmakers, Mark van Staalduinen

TNO

Brassersplein 2, 2612 CT, Delft, The Netherlands

Wessel Kraaij

TNO & Radboud University Nijmegen

E-mail of corresponding author: john.schavemaker@tno.nl

1. STRUCTURED ABSTRACT

The TNO instance search submission to TRECVID 2010 consisted of three different runs, two using a bag-of-visual-words approaches and one using a commercial face-recognition software package.

Briefly, what approach or combination of approaches did you test in each of your submitted runs?

- all runs: video decoding using ffmpeg library, sampling every 25th frame.
- **F_X_NO_TNO_1**: standard SURF keypoint detection, bag-of-words using 4096 prototypes from videos, indexing and querying using Lemur.
- **F_X_NO_TNO_2**: standard SURF keypoint detection, bag-of-words using 256 prototypes from queries, indexing and querying using Lemur.
- **F_X_NO_TNO_3**: commercial face-detection package.

What if any significant differences (in terms of what measures) did you find among the runs?

In terms of average precision summed over all queries TNO run 2 significantly outperforms TNO runs 1 and 3. Runs 1 and 3 do not differ significantly in terms of this summed average precision.

Based on the results, can you estimate the relative contribution of each component of your system/approach to its effectiveness?

The results of TNO run 3 show that this commercial face-detection package with default settings has no significant contribution to effectiveness. Based on the difference between TNO runs 1 and 2 we can estimate that the relative contribution of choosing a small visual vocabulary build on the query set of images is high.

Overall, what did you learn about runs/approaches and the research question(s) that motivated them?

What we learned from our runs: using a commercial face-detection package without tweaking on this (low image quality) dataset does not work. Using a small-sized (512 words) visual vocabulary computed on the query set significantly outperforms a much larger (4096 words) visual vocabulary on the whole dataset. One can build an image-retrieval system using open-source components.

2. INTRODUCTION

In this notebook paper we describe our approaches to the TRECVID 2010 instance search tasks and analyze the results of our submissions. TNO has submitted three runs: two runs using a bag-of-visual-words approach and one run using a commercial-off-the-shelf (COTS) face-recognition software package. The main rationale behind all three runs was: "Can we build an instance-search system from scratch using only open source or commercial components without significant algorithmic development of our own?" The remainder of this notebook paper is as follows. The paper starts with a short section on data analysis of the video and query data set in Section 3. In Section 4 we describe in detail the processing steps of the three runs and their software implementation. In Section 5 we present some of the results of the three runs and compare them. In Section 6 we discuss some of the observations we have made on the video and query data as well as the chosen algorithms for the different runs.

3. Data analysis

3.1 Video data set

The video data set consists of 400 MPEG-1 videos from the Sound & Vision archives, some general features we noticed:

- videos in color and black-and-white;
- all videos in 352x288 video resolution;
- different recordings from Sound and Vision archive;
- some videos have subtitling in Dutch.

3.2 Query data set

The query set consists of 22 queries (cf Figure 1). For every query multiple query images are given up to five images per query (e.g. Figure 2). And for every query image five formats are given: source image (full video frame), target image (rectangular region-of-interest in video frame), mask image (binary segmentation for target), object image (segmented target image), and outline (segmented target image with mask contour depicted in red) (cf. Figure 3). The query images and target video have the following properties:

- all query images in color with exception of images 9006.3 – 9006.5;
- different source video resolutions: 640x480 and 352x288;
- different target regions resolutions: 32x13 up to 626x323.

The query dataset contains four types of queries: persons, characters, objects and locations. Face recognition was expected to be relevant for the first two categories, which contain 8 and 5 queries, respectively. Of course, we expected some confusion in the result set for the character queries, since one actor may appear as multiple characters in the video data set.

The image data of HRH Prince Bernhard (9006) is particular interesting, since some query images were shot in the 1940s in black and white, while other images dated from around the year 2000.

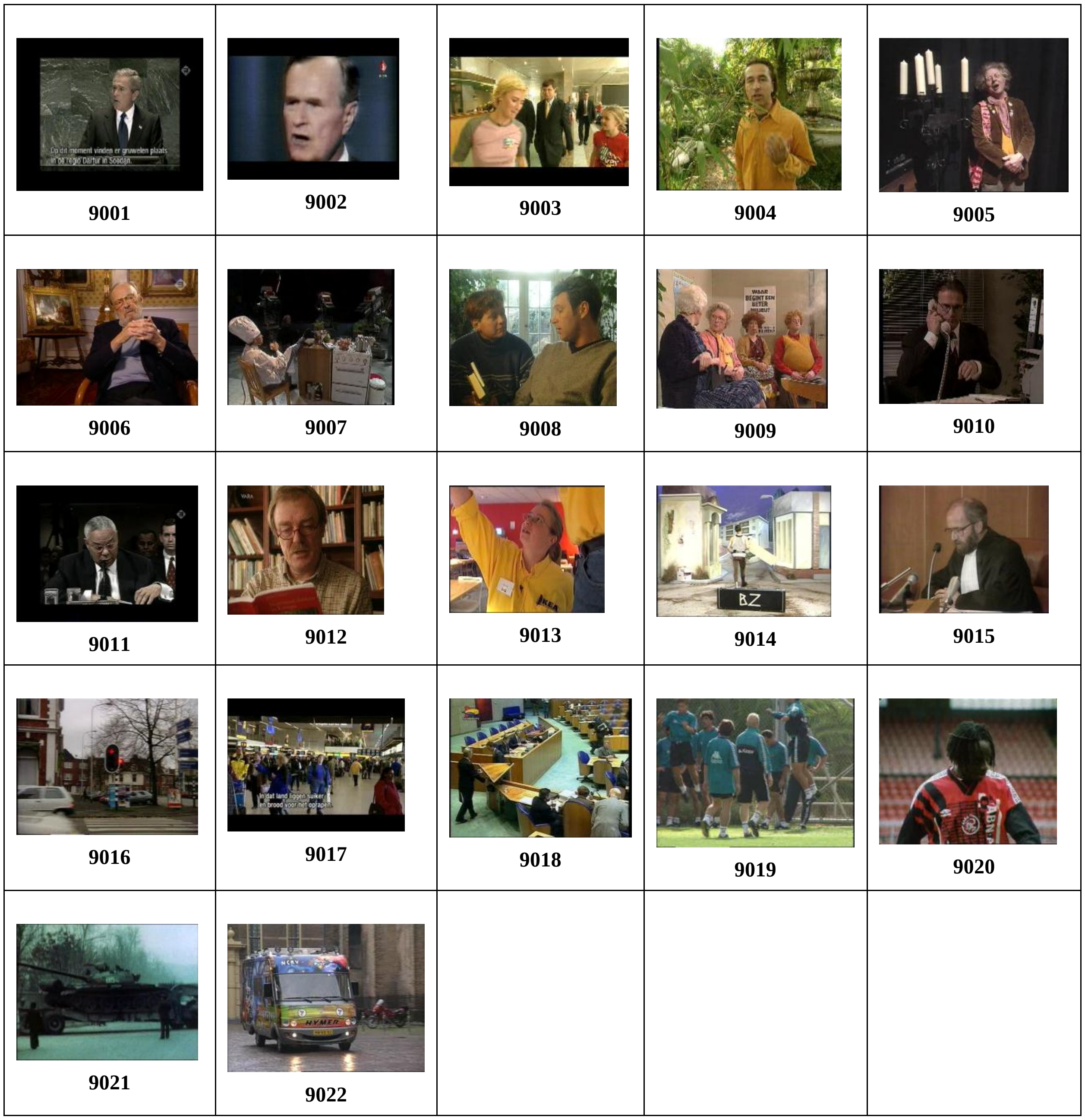


Figure 1: first sample image of each of the 22 queries

9022.1	**9022.2**	**9022.3**	**9022.4**	**9022.5**

Figure 2: Five images for a query

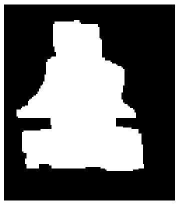
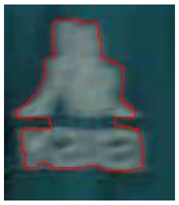
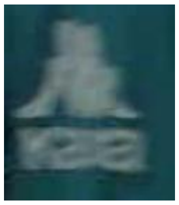
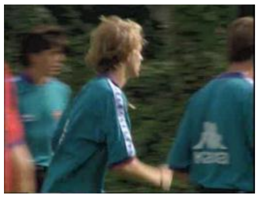
 **mask**	**object**	 **outline**	 **Target**	 **src**

Figure 3: Mask, object, outline, target and source image

4. Approach

The chosen approach differs from TNO runs 1, 2 and 3 so they are described separately in different sections:

- **F_X_NO_TNO_1**: data-based bag-of-visual-words retrieval
- **F_X_NO_TNO_2**: query-based bag-of-visual-words retrieval
- **F_X_NO_TNO_3**: commercial face-detection package

For all runs the pre-processing (decoding and framing) of the videos in the dataset was the same. For video decoding we used the FFmpeg library [1] as integral part of the Open Computer Vision Library [2] and stored every 25th frame as a JPEG image. The 1Hz sampling was chosen because of storage size and processing time considerations.

4.1 Run 1 approach: data-based bag-of-visual-words

The first run uses the classical bag-of-visual-words approach with some minor differences. The processing steps in detail:

1. SURF key points are detected in every decoded and stored video frame of the entire video data set.
    a. chosen implementation: cvExtractSurf from OpenCV [2,6] using default parameters;
    b. key point detection is sparse and based on the Hessian structure tensor, differing here from the classical dense-sampled approach using a pre-defined grid of points with fixed scales;
    c. SURF descriptors are computed (128 floating point numbers) and stored with key point information in XML.
2. A subset of all SURF key point descriptors from the entire data set are clustered to a pre-defined number (here 4096 cluster prototypes)
    a. chosen implementation: flann::hierachicalClustering from FLANN[3] in OpenCV [2] using default parameters;
    b. subset selection is simply done by sampling again, taking approximately the key points of every 500th stored video frame (depending on memory);
    c. the visual vocabulary cluster prototypes are written to XML.
3. The set of SURF key points from every stored video frame of the entire video data set is quantized according to visual vocabulary from step 2.
    a. chosen implementation: flann::knnSearch from FLANN[3,7] in OpenCV [2] using default parameters;
    b. the quantization vector is converted to text in TRECTEXT format (see [4]), where every visual word and its occurrence is encoded. Encoding in text is done by using words “w0” to “w4095” and repeating the same word for multiple occurrences of it in the quantization vector. The DOCNO tag encodes the video and frame number:

```
<DOC>


<DOCNO>BG_1387_mpg_item5_sample0<
/DOCNO>


<TEXT>


w37 w46 w73 w127 w150 w180 w246
w263 w265 w289 w430 w500 w500
```

```
w584 w589 w589 w593 w659 w659
w666 w741 w760 w852 w854 w854
w854 w915 w915 w915 w917 w1014
w1016 w1038 w1094 w1110 w1150
w1153 w1153 w1153 w1284 w1295
w1376 w1507 w1510 w1510 w1510
w1512 w1524 w1525 w1530 w1565
w1636 w1637 w1792 w1878 w1885
w1942 w1996 w2018 w2019 w2030
w2033 w2184 w2276 w2347 w2387
w2387 w2389 w2851 w2891 w2955
w2958 w3042 w3042 w3354 w3472
w3593 w3594 w3595 w3647 w3647
w3676 w3780 w3829 w3961 w3961
w3963 w3967


</TEXT>


</DOC>
```

4. A Lemur repository is created from all TRECTEXT formatted documents that contain a quantization of the SURF key point descriptors using the visual vocabulary.
   a. Using the C++ indri::api::create() and addFile() methods with default parameter settings.
5. For a query with multiple images steps 1 and 3 are alike for the set of query images:
   a. SURF key points are detected and corresponding descriptors are computed from the query image (we use the segmented “object” image);
   b. For every query image the TRECTEXT quantization string is determined from the quantized SURF descriptors in the “object” region of the image;
   c. With the TRECTEXT quantization string a Lemur query is done for every query object image using C++ indri::api::QueryEnvironment::RunQuery() asking for 1.000 results. Every result consists of a video ID (e.g. BG_1387_mpg) and a video frame number in that specific video.
   d. The results for the different query images (usually five) are put together and sorted against the Lemur index score that is computed for every result.
   e. For every result (frame in video) the corresponding shot ID is retrieved; duplicate shots are removed.
   f. The top 1000 shots are kept and written to TRECVID XML output.

4.2 RUN 2 APPROACH: QUERY-BASED BAG-OF-VISUAL-WORDS

The approach of run 2 is the same as for run 1 except that the visual vocabulary used is not calculated on the video data set but on the query images set of size 22 times 5 “source” images. The size of the visual vocabulary in run 2 is 512 words (instead of the 4096 words of run 1). The idea of this run was how a relatively small but specialized vocabulary would work at the cost of re-quantizing your original data set with the query images.

4.3 RUN 3 APPROACH: COMMERCIAL FACE-DETECTION PACKAGE

The core of the third run is a commercial face-recognition package. The processing steps in detail:

1. The actual faces are first detected in the images using the face detection algorithms in OpenCV. When a face is detected, the image is cropped such that only the detected face and some margin is preserved, and saved to a JPEG file.
2. For each facial image, the face recognition algorithm computes a proprietary description. When no face is detected by the package, no output is generated.
3. Each description of each facial image in the query dataset is compared to each facial image in the video dataset. The output is a ranked list with a confidence score per input image.
4. Since multiple input images are available per query, the result sets for the images belonging to the same query are merged and sorted. The frame numbers are converted to a shot index using the timing information in the TRECVID shot list.
5. The individual items in the list belonging to the same shot are merged into a single item. The aggregated confidence score c_{shot} is obtained from the frame level confidence scores c_{frame} by:

$$c_{shot}(n) = \max_{\substack{\text{frames } i \text{ in} \\ \text{shot } n}} \left(c_{frame}(i)\right) * \sum_{\substack{\text{frames } i \text{ in} \\ \text{shot } n}} c_{frame}(i)$$

The face recognition package (steps 2 and 3) is treated as a black-box. Only the result of the matching is post-processed (steps 4 and 5).

Face recognition is typically used in biometrical systems, but the last decade face recognition is introduced in surveillance cameras, games consoles and consumer software. Performance is highly dependent on lighting conditions, the angle at which the facial image is acquired, the quality of the images. Many applications are based on verification (1-to-1 comparison) in contrast to identification (1-to-N comparison). Consumer applications such as Picasa and iPhoto include face recognition algorithms, but put the user in the loop to annotate the videos and interactively improve the recognition result. Since we did not touch the matching algorithm, and performed a fully automatic identification, the system did not learn from a cluster of facial data, but considered each face individually.

A critical factor for face recognition is the distance between the eyes in the facial image. Typically the minimum inter-pupil distance between the eyes should be 32-64 pixels, and poses may deviate up to 15°-20°. Many of the query images do not satisfy these criteria.

5. RESULTS

In this section we present some of the results our runs made on the instance search task. In the first figure we show an overall ranking of the submissions of all participants and indicated the relative position of the different TNO runs. The total precision of each participant is computed as the sum of the precision of all 22 queries. In this figure we have left out the participants using an interactive run approach to have a better scaled graph and comparison.

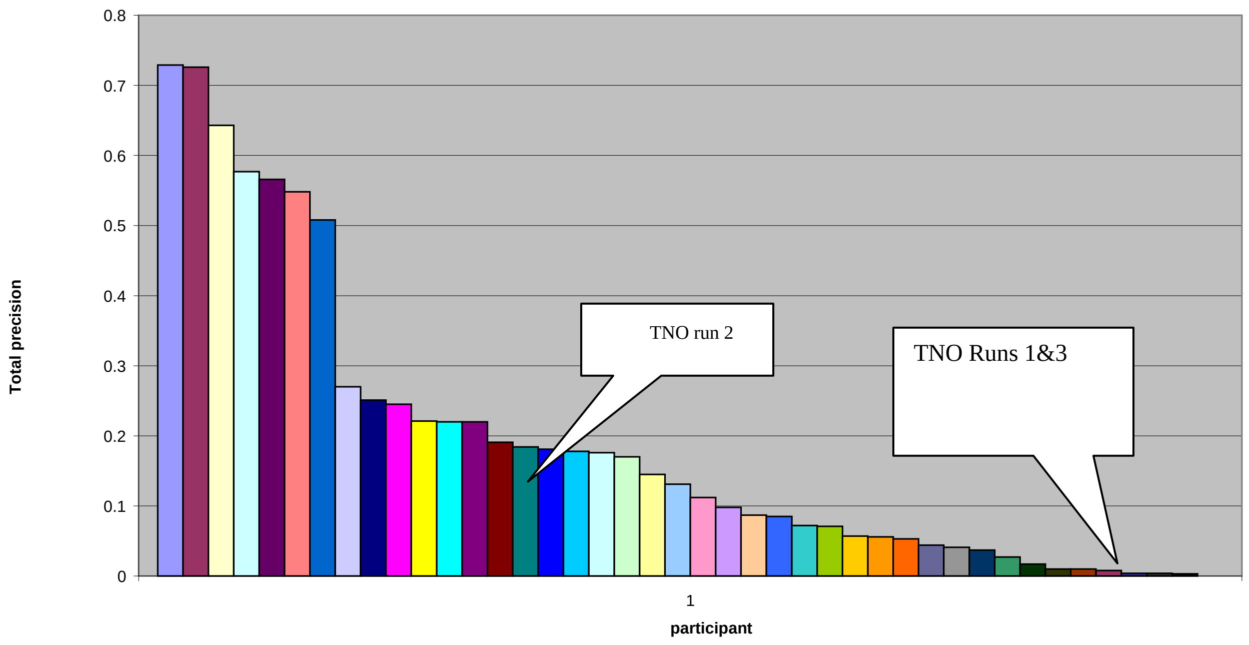


Figure 4: overview of all submitted runs

From Figure 4 it is clear that TNO runs 1 & 3 did not perform well w.r.t. the submission of other participants. TNO run 2 shows more than average performance.

Figure 5 shows the performance per query. In order to see the difference a log(1000 * precision) measure is used. The figure chart also indicates the best, median, and bottom performance from the previous figure to illustrate relative performance of the TNO runs. TNO run 3 (face recognition) is close (or equal) to the bottom performance. TNO run 1 shows some precision for queries 9007 and 9009. TNO run 2 shows for about 7 queries some precision of which 5 are better than the median performance and 4 are better than the top performance.

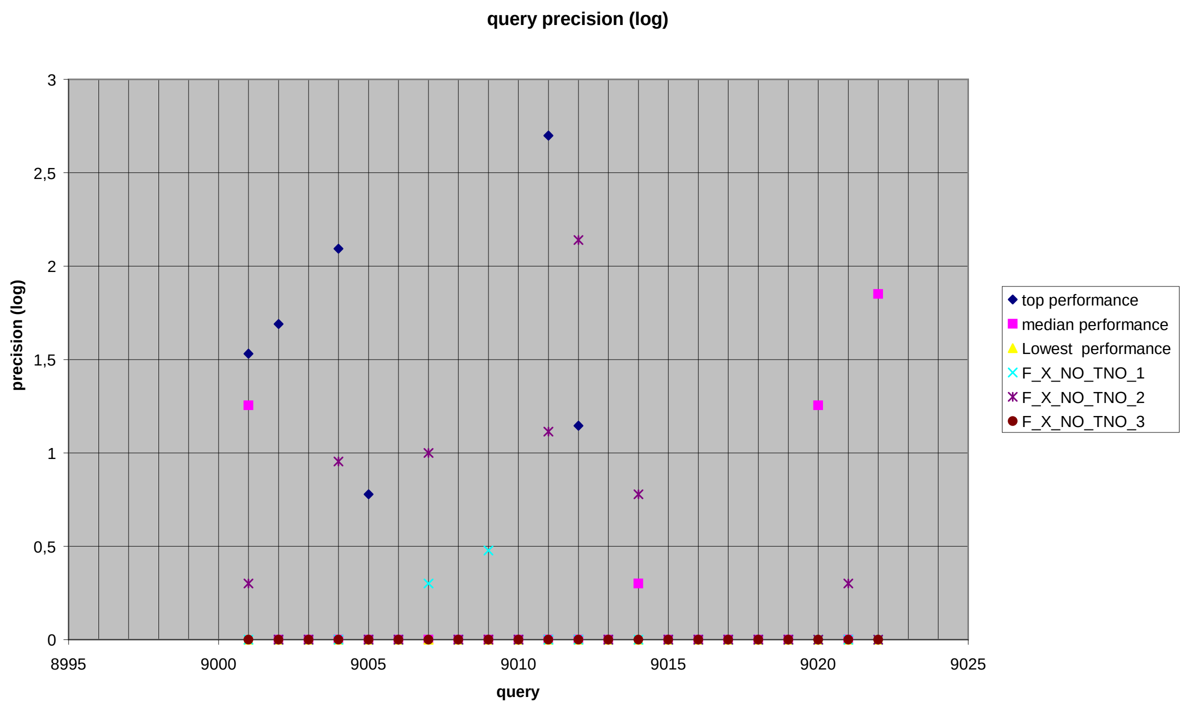


Figure 5: log mean average precision per query

6. OBSERVATIONS

In this Section we elaborate and discuss some observations from the result and our experiences with the instance search task using the video and query data sets.

- **Subtitling**:
  - Some videos have subtitling, with commercial or open-source OCR software (with some adaption to video: font choice and text location at bottom) one can read that text and store it with as extracted metadata. Some of the queries contain descriptions (‘President Bush’) that could be used in this way. In our runs we did not apply this (sub)approach but technically it is interesting (sub) approach if metadata is available for the query.
  - Subtitling attracts a lot of detected key points (in our standard, sparse key-point detection approach). This could potentially influence the determination of the visual words in our approach: some portion of the vocabulary may be of subtitling origin. We noticed this effect when a query image contained subtitling (for example the source image of query 9001.1); the top results showed a lot of video frames containing text only (like the credits at the end of a movie).
- **Sampling:**
  - In our run approaches we have sampled the video data sets by selecting one frame per 25 video frames (roughly 1 frame per second). This choice was motivated by pragmatic estimates on the required disk storage for the decoded video frames and associated metadata as key points and descriptors in XML. Furthermore, this choice limits processing time to be able to make algorithm iterations possible in our (short) time frame. The pre-processed video data set now roughly requires 500 Gigabyte disk storage. Processing steps for the whole video data sets like key point detection, key point quantization, building a visual vocabulary, etc. take in the order of hours (using parallel processing on frames) on a single 8-core PC.
  - Using sampling the probability that you find the exact video frame is of course 1 out of 25. Our idea was that persons, objects, and locations are longer in view than a single frame and a key point approach can deal with changes of viewpoint so you do not loose much sensitivity in this way. We have not evaluated our results on this aspect.
- **Bag-of-words**:
  - The difference between TNO runs 1 and 2 was how the visual vocabulary was build. Run 1 used all video data with a size of 4096 words, run 2 used only the query set of images with a vocabulary size of 512 words. Run 2 performed significantly better than run 2, so it is in principle to have some query precision using a small, specialized vocabulary but one needs to re-quantize the whole data set with the query set at hand. That may prove not to be a practical solution for larger datasets.
  - We observed that TNO run 1 had good generalization properties and acted some times as a concept detector: query 9001 “President Bush” resulted in a lot of “similar” hits of men in dark suits with tie. Unfortunately, it seems that with this generalization some more specific features of this object are lost (on not represented in the visual

vocabulary) because the real “instances” did not float to the surface of results.

7. ACKNOWLEDGEMENTS

The work described in this paper was supported by the Dutch national programme "Recognition of Digital Information and Fingerprinting".

8. REFERENCES

[1] http://ffmpeg.org/

[2] http://sourceforge.net/projects/opencvlibrary/

[3] http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN

[4] http://www.lemurproject.org/

[5] 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

[6] Herbert Bay, Tinne Tuytelaars and Luc Van Gool *"SURF: Speeded Up Robust Features"*, Proceedings of the 9th European Conference on Computer Vision, Springer LNCS volume 3951, part 1, pp 404--417, 2006

[7] Marius Muja and David G. Lowe, *"Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration"*, in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009