Notebook paper: TNO instance search submission 2011

John Schavemaker, Pieter Eendebak, Mark van Staalduinen, Wessel Kraaij
TNO Technical Sciences
Brassersplein 2, 2612 CT, Delft, The Netherlands
E-mail of corresponding author: john.schavemaker@tno.nl

Structured Abstract

The TNO instance search submission to TRECVID 2011 consisted of three different runs: one is using an exhaustive keypoint search, one is using a bag-of-visual-words approach and one is using open-source face-recognition software. Our run approaches:

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-SURFAC2_1**: standard SURF keypoint detection, exhaustive search.
- **F_X_NO_TNO-BOWCOL_2**: standard SURF keypoint detection, bag-of-words using 256 prototypes from queries, indexing and querying using Lemur.
- **F_X_NO_TNO-SUREIG_3**: face detection & recognition with Eigenface descriptors.

What if any significant differences (in terms of what measures) did you find among the runs?
In terms of average precision TNO run 1 significantly outperforms TNO run 2. Runs 1 and 3 differ only at the PERSON queries.

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 face-recognition software can have a contribution to effectiveness for PERSON queries. Based on the difference between TNO runs 1 and 2 we can estimate that the contribution of choosing exhaustive keypoint search over bag-of-words with 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 face-recognition software on this dataset can have a contribution if the queries contain large frontal faces. Exhaustive keypoint search significantly outperforms bag-of-words. One can build an image-retrieval system using open-source components.

Introduction

In this notebook paper we describe our approaches to the TRECVID 2011 instance search tasks and analyze the results of our submissions. TNO has submitted three runs: one is using an exhaustive SURF keypoint search, one is using a bag-of-visual-words approach and one is using open-source face-recognition software. 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 "Data Analysis". In Section "Approach" we describe in detail the processing steps of the three runs and their software implementation. In Section "Results" we present some of the results of the three runs and compare them. In Section "Conclusions" we discuss some of the chosen algorithms for the different runs.

Data analysis

Video data set

The video data set consists of about 20000 video rushes from the BBC archives, some general features we noticed:

- videos mostly in color;
- videos in different resolutions;
- short videos, about 2 minutes;
- no subtitling.

Query data set

The query set consists of 25 queries. For every query multiple query images are given up to six images per query. And for every query image two formats are given: source image (full video frame) and mask image (binary segmentation for target), all query images in color and with image resolution: 352x288.

The query dataset contains three types of queries: persons, objects and locations. Face recognition was expected to be relevant for the first category, which contains 6 queries.

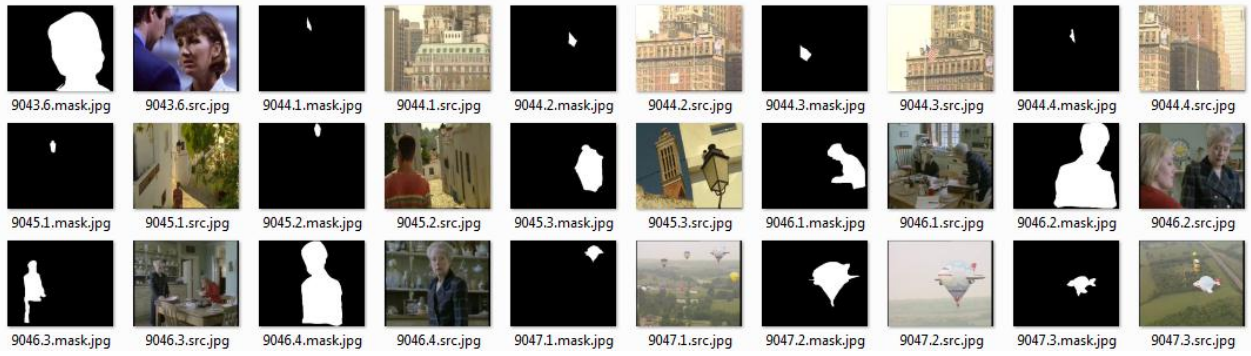


Figure 1: Example mask and source images from the query image set.

Figure 2: Five image for one query.

Figure 3: Mask and source image for one query image.

Approach

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

- **F_X_NO_TNO-SURFAC2_1**: retrieval by object recognition
- **F_X_NO_TNO-BOWCOL_2**: query-based bag-of-visual-words retrieval
- **F_X_NO_TNO-SUREIG_3**: open-source face-recognition software

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.

Run 1 approach: object recognition

Our first approach is interesting because it is the original object recognition idea outlined by David Lowe in his paper on SIFT [7]. In our approach we follow the same scheme of detecting key points, computing and matching local descriptors but instead of a few objects in the recognition database we have descriptors from every video. We handle that large amount of descriptors on an application level by dividing the resulting descriptor 'matrix' in a number of binary files and process them one by one.

In our experiments we have chosen SURF [6] over SIFT because of processing time considerations. As with the approach of Lowe, key point detection is done sparse and is not using a sampling with a pre-defined grid of points. We use the default SURF detector based on the Hessian structure tensor. Descriptor matching is done with an approximate nearest-neighbor

implementation where only a part of all descriptors can be in memory. This approach showed more than average (median) results for all queries, for two it was the best performing one.

Run 2 approach: query-based bag-of-visual-words

Our third approach used bag-of-words extended with face detection and color quantization. It showed good generalization results but it seems less suited for instance search. The interesting part is that all visual features are converted to text and that a text-retrieval framework can be used for quick (interactive) searching. Furthermore, the vocabulary is computed on the query image set instead of the videos.

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] 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. All SURF key point descriptors from the query image set are clustered to a pre-defined number (here 256 cluster prototypes)
    a. chosen implementation: flann::hierachicalClustering from FLANN[3] in OpenCV [2] using default parameters;
    b. 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] 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 “w255” and repeating the same word for multiple occurrences of it in the quantization vector. The DOCNO tag encodes the video and frame number;
4. Quantization of global color in a video frame:
    c. find black (low value) and white (high value, no saturation) pixels;
    d. make global color HSV histogram (9 bins) of remaining pixels;
    e. quantize bin (color) with scheme (for example “LowYellow”) and add to text:
        i. more than 1% “Low”,
        ii. more than 10% “Medium”,
        iii. more than 25% “High”
5. Quantization of face detection: face detected add “face” to TRECTEXT text:

```
<DOC>

<DOCNO>1000_mpg_item3_sample2</DOCNO>

<TEXT>

w5 w6 w7 w8 w9 w17 w19 w20 w21 w22 w24 w25 w26 w27 w28 w29 w30 w34 w42
w51 w52 w59 w60 w62 w64 w66 w67 w68 w69 w71 w73 w76 w77 w78 w81 w84 w86
w89 w90 w96 w97 w113 w114 w117 w120 w130 w171 w177 w180 w181 w182 w189
w192 w196 w197 w203 w206 w207 w222 w224 w241 w249

face LowOrange MediumOrange HighOrange LowYellow MediumYellow LowBlack

</TEXT>

</DOC>
```

6. A Lemur repository is created from all TRECTEXT formatted documents that contain a quantization of the SURF key point descriptors using the visual vocabulary.
    f. Using the C++ indri::api::create() and addFile() methods with default parameter settings.
7. For a query with multiple images steps 1, 3, 4 and 3 are alike for the set of query images:
    g. SURF key points are detected and corresponding descriptors are computed from the query image, color is computed, and faces detected.
    h. For every query image the TRECTEXT quantization string is determined from the quantized SURF descriptors, computed color histogram and detected faces;
    i. 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. 1000_mpg) and a video frame number in that specific video.
    j. 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.
    k. For every result (frame in video) the corresponding shot ID is retrieved; duplicate shots are removed.
    l. The top 1000 shots are kept and written to TRECVID XML output.

Run 3 approach: open-source face-recognition software

In our third approach we used face recognition for the PERSON queries, for the other queries we used the first approach. For one of the PERSON queries with good frontal faces in the query set we received a 7th best position. Our approach is based on the Viola & Jones face detection and simple Eigenface descriptors [9]. Matching is done as with the SURF descriptors but with all descriptors in memory and giving instant search results.

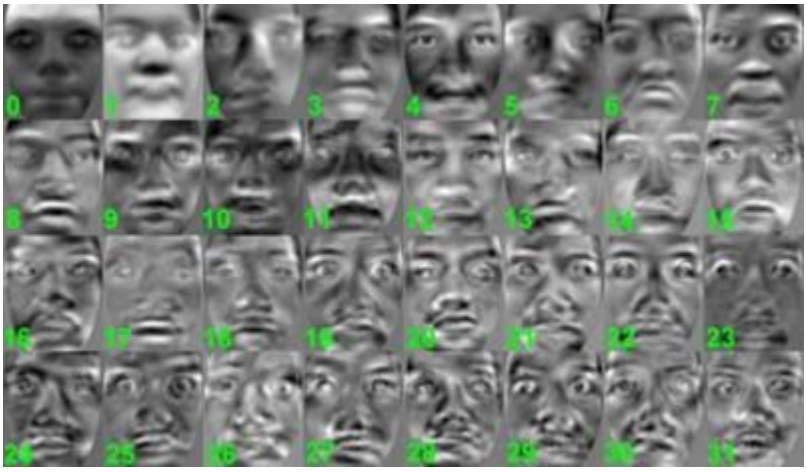


Figure 4: Eigenfaces.

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 an Eigenface description. When no face is detected, 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.

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 tune 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.

Results

In this section we present some of the results our runs made on the instance search task. In the first three figures we show the average precision of our three runs versus the median and best scores by topic.

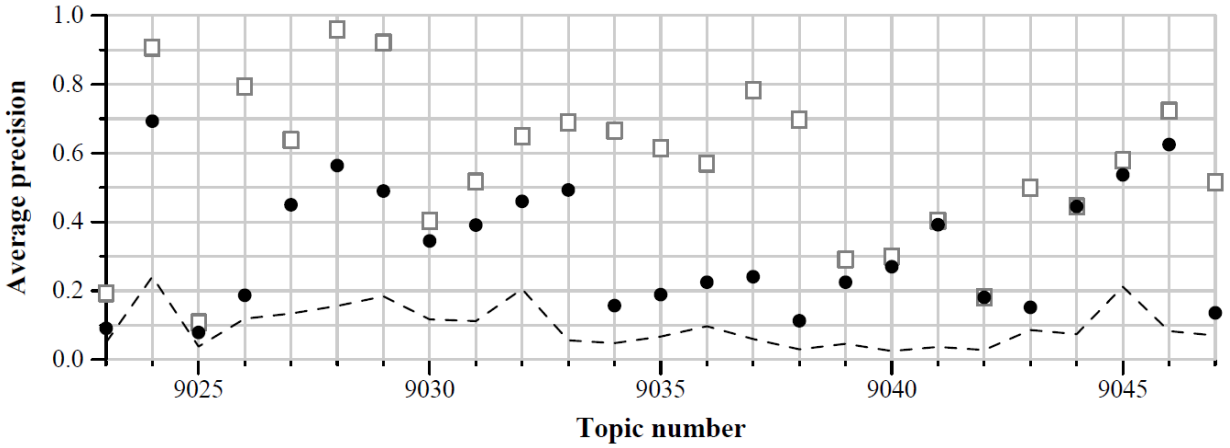


Figure 5: Average precision of run F_X_NO_TNO-SURFAC2_1 (dot) versus median (---) versus best (box) by topic.

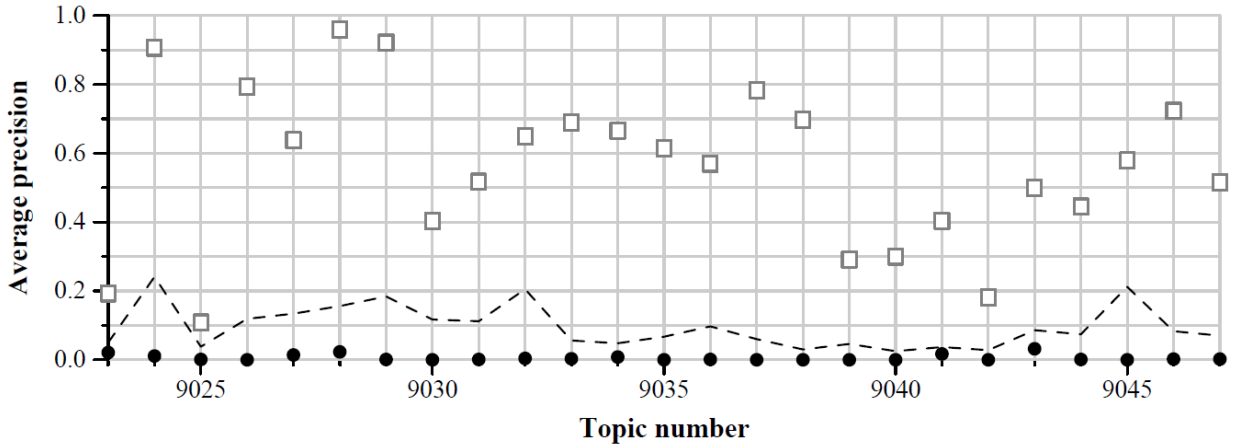


Figure 6: Average precision of run F_X_NO_TNO-BOWCOL_2 (dot) versus median (---) versus best (box) by topic.

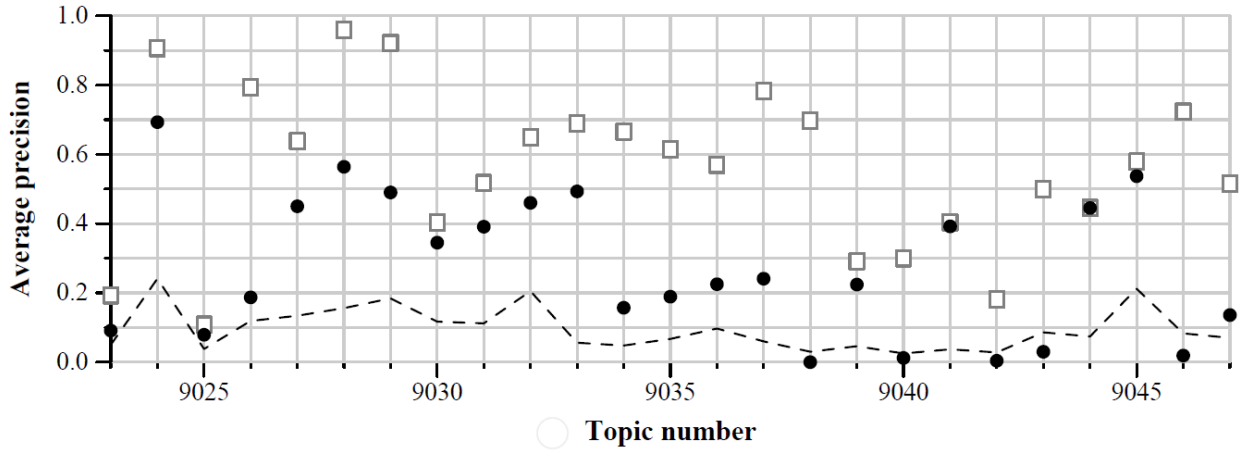
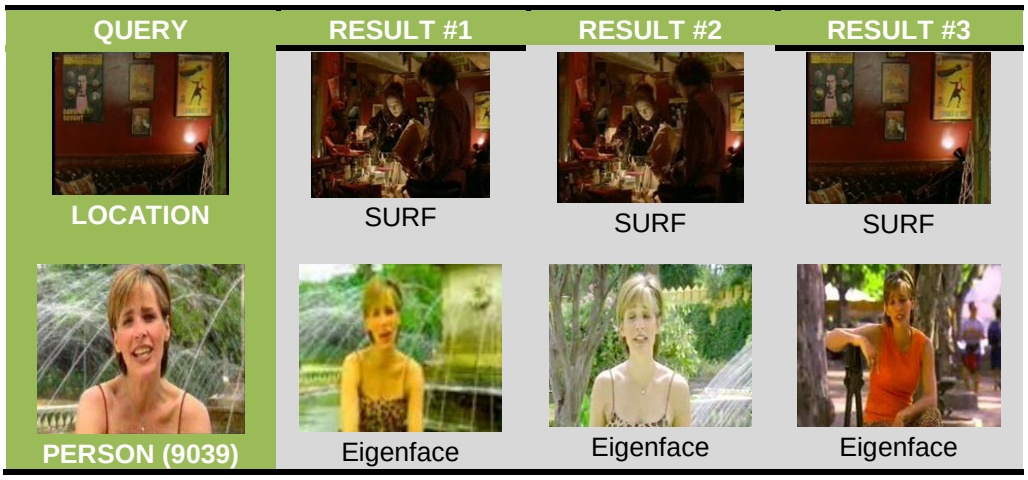


Figure 7: Average precision of run F_X_NO_TNO-SUREIG_3 (dot) versus median (---) versus best (box) by topic.

From these figures it is clear that TNO run 1 performs best w.r.t. to the other two runs and average performance, for two topics it was the best performing one. For one of the PERSON queries with good frontal faces in the query set (9039) we received a 7th best position for TNO run 2. TNO run 3 (bag-of-words) did not perform as expected.



Conclusions

(SURFAC2_1) Using a brute-force search over all SURF keypoint descriptors gives real instance search results: hits for every query. However, there is no generalization (for example, sunset) of objects and it takes relatively long (15-30 minutes on a single-core PC).

(BOWCOL_2) Bag-of-words gives for some queries nice concept detection results (sunset) and is relatively fast.

(SUREIG_3) Some interesting results for searching persons. With good frontal faces in the query set, interesting results are possible and quick to attain.

Acknowledgements

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

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, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008
[7] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157.
[8] L. Sirovich and M. Kirby (1987). Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America A, 4, 519–524.
[9] Robin Hewitt, “Seeing With OpenCV, Part 5: Implementing Eigenface”, http://www.cognotics.com/opencv/servo_2007_series/part_5/index.html.