Notebook paper: TNO instance search submission 2012

John Schavemaker, Corné Versloot, Joost de Wit, 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 2012 consisted of four different runs: two are using an exhaustive SIFT key-point search, and two are using an extended bag-of-visual-words approach. 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-ANNMSK_1**: standard SIFT key point detection, exhaustive search using FLANN matching with query key points inside query mask.
- **F_X_NO_TNO-ANNMSK_3**: standard SIFT key point detection, exhaustive search using FLANN matching with query key points inside query mask and resolution of query images enhanced by a factor 2 for small objects.
- **F_X_NO_TNO-INOMSK_2**: standard SIFT key point detection within mask region, extended bag-of-words using 1000 prototypes from videos, indexing and querying using Lucene.
- **F_X_NO_TNO-INORDR_4**: standard SIFT key point detection, extended bag-of-words using 1000 prototypes from videos, indexing and querying using Lucene.

What if any significant differences (in terms of what measures) did you find among the runs?
In terms of average precision TNO runs 1 and 2 significantly outperform TNO run 3 and 4. TNO run 2 with resolution enhancement performs slightly better than run 1 without enhancement.

Based on the results, can you estimate the relative contribution of each component of your system/approach to its effectiveness?
Based on the difference between TNO runs 1 & 2 and 3 & 4 we can estimate that the contribution of choosing exhaustive key-point search over bag-of-words with a small visual vocabulary is high. The contribution of resolution enhancement is low.

Overall, what did you learn about runs/approaches and the research question(s) that motivated them?
Exhaustive key point search significantly outperforms bag-of-words at the cost of extra query time. One can build an image-retrieval system using open-source components.

Introduction

In this notebook paper we describe our approaches to the TRECVID 2012 instance search tasks and analyze the results of our submissions. TNO has submitted four runs: two are using an exhaustive SIFT key point search using FLANN matching, and two are using a bag-of-visual-words approach.

The main rationale behind all four 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 four 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 70000 internet videos in WebM format, downloaded from Flickr under Creative Commons licenses, some general features we noticed:

- videos mostly in color;
- videos in different resolutions;
- very short videos, at maximum a few minutes;
- some videos contain no video frames;
- no subtitling.

Query data set

The query set consists of 21 queries. For every query multiple query images are given up to nine images per query. And for every query image two formats are given: source image (full-sized image) and mask image (binary segmentation for target object), all query images in color and with image resolutions ranging from: 352x240 to 640x480.

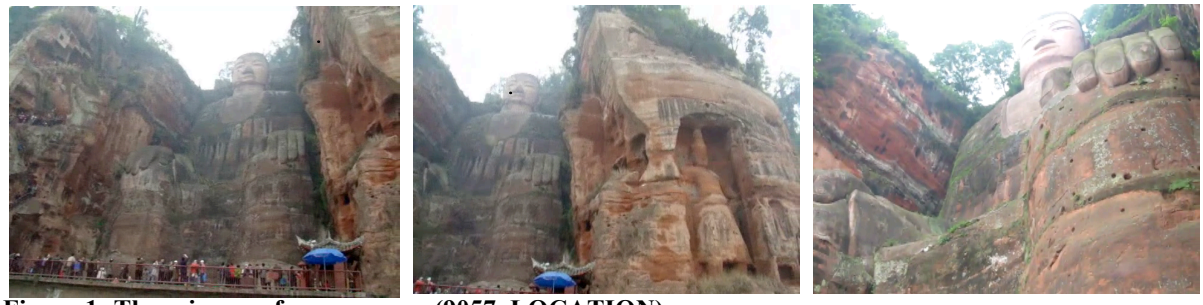


Figure 1: Three images for one query (9057, LOCATION).

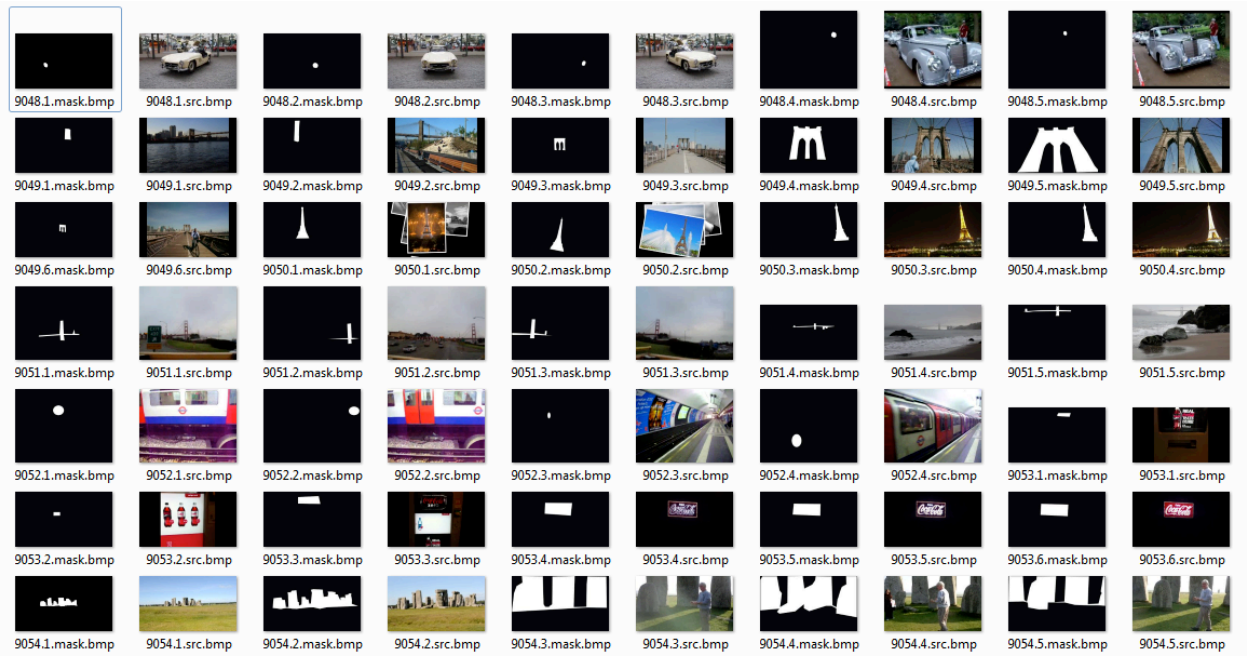


Figure 2: Example mask and source images from the query image set.

The query dataset contains three types of queries: persons, objects and locations. For example query 9060 is a person query, 9048 an object query and 9057 a location query.

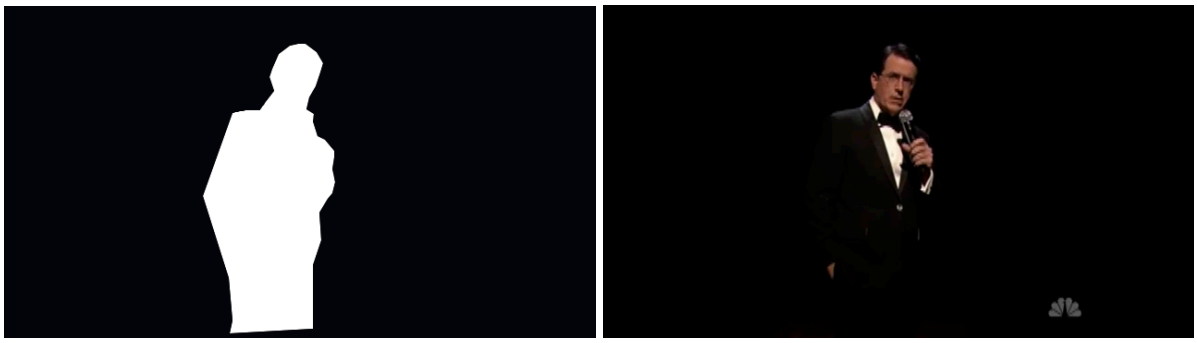


Figure 3: Mask and source image for one query image (9060, PERSON).

Approach

The chosen approach differs from TNO runs 1 & 2 and 3 & 4 so they are described separately in different sections:

- **F_X_NO_TNO-ANN***: retrieval by object recognition (runs 1 & 2)
- **F_X_NO_TNO-INO***: bag-of-visual-words retrieval (runs 3 & 4)

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.

Runs 1 and 2 approach: object recognition

Our first approach for runs 1 and 2 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 (in the order of hundreds, depending on memory) and process them one by one. Multiple queries can be combined within one single run.

In contradiction with our earlier experiments in 2010 and 2011 we have chosen SIFT over SURF [6]. 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 SIFT detector based on difference of Gaussians. Descriptor matching is done with an approximate nearest-neighbor implementation (FLANN) where only a part of all descriptors can be in memory. This approach showed more than average (median) results for all queries, for one (9057) it was the best performing one.

The difference between run 1 and 2 is that for run 2 the image resolution for small objects (determined by mask area) is enhanced by a factor 2 to generate more detected key points at the cost of possibly less optimal descriptors. The resolution enhancement has a small positive effect on the results.

Run 3 and 4 approach: extended bag-of-visual-words

Our 2nd approach (runs 3 and 4) uses an extended bag-of-words. 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 videos.

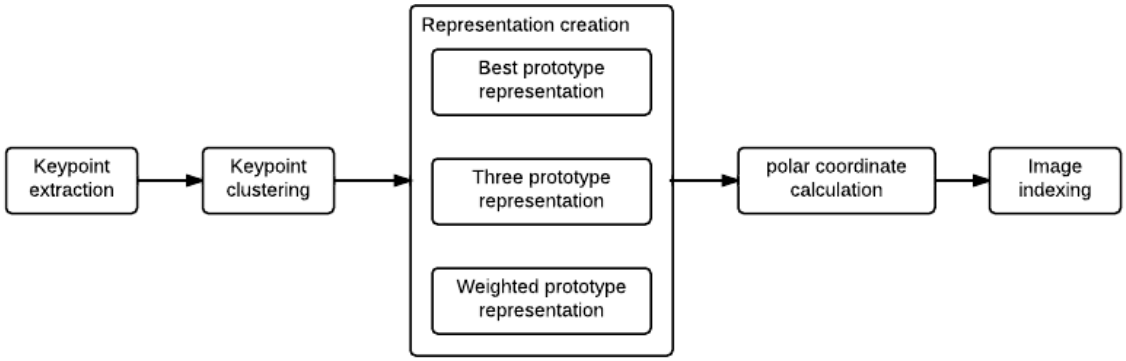
With the extended Bag of Words(EBoW) approach we tried to improve matching quality while keeping the fast matching speed typical for BoW methods. The EBoW method differs from the normal approach in two ways:

- Using a variety of textual representations, some enable uses of wildcards for flexible matching
- Use of the coordinates of the key points in the scoring function.

The indexing and search processes followed are described in detail in the sections below. Lucene was used as the search engine.

Indexing

The EBoW method is based on text search and hence consists of two major steps: indexing and searching. The indexing pipeline used to analyze and store a single image is shown in the figure below.



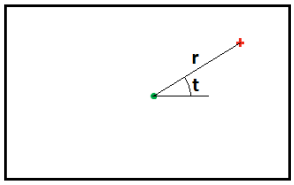
First key points are extracted and clustered resulting in 1000 prototypes. For each key point the three 'nearest' prototypes (including the distance of the key point to each prototype centroid) are used. Each key point is represented as: *proto385-0.03412_proto041-0.00725_proto795-0.00732* (prototype term followed by the distance of the key point to that prototype centroid).
In the actual text-indexing phase these long strings are represented in three different ways:

1. Best prototype: only the best prototype is stored, in the example above this would be 'proto385' (classic BoW approach)
2. Prototypes: thee prototypes are stored in order without their distances: '*proto385_proto041_proto795*'
3. Weighted prototypes: the prototypes are stored including their distance values: 'proto385-0.03412_proto041-0.00725_proto795-0.00732'

Each key point in an image is transformed to these three representations that are all stored in their own fields (concatenated using a single space) resulting in a text document as shown below.

Image: 9048_1.jpg
Best_proto: proto385 proto068 proto001 etc.
Prototypes: proto385_proto041_proto795 proto068_proto156_proto985 etc
Weighted_proto: proto385-0.03412_proto041-0.00725_proto795-0.00732 proto068-0.1250_proto156-0.0567_proto985-0.0086 etc.

The polar coordinate (t,r) of each key point in the image is calculated with respect to the center of the image and normalized using the images height and width. Each representation is stored in the index along with its polar coordinate as payload. This payload is used in the search process to change the score of an indexed image with respect to a query image.



Searching

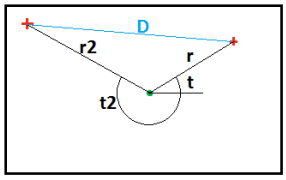
The search process is very similar to the indexing process: key points are extracted, clustered and representations created resulting in a text document as shown before. This text document forms the basis for the query to be executed. There are many different search strategies possible:

1. Use one field only (best_proto, prototypes and weighted_proto)
2. Use multiple fields, possibly boosting one field with respect to the other(s)
3. Use wildcards within the weighted_proto queries to control the precision during search. For example a query such as proto385-0.03???_proto041-0.00???_proto795-0.00??? will match more precisely than proto385-0.0????_proto041-0.0????_proto795-0.0????
4. Weight of the coordinates of matched key points with respect to the coordinates of the query key points. The role of the coordinates is described in a separate section below

Different search strategies use one or multiple of the three fields (best_proto, prototypes and weighted_proto) as a very large OR query. It is possible to boost one type of field with respect to the other(s). An example query is shown below where *prototypes* field is boosted 10 times with respect to the *best_proto* field (results based on the *prototypes* field contribute 10 times more to the relevance score than results from *best_proto* do).

Best_proto:(proto385 OR proto068 OR proto001 OR …) OR
Prototypes:(proto385_proto041_proto795 OR proto068_proto156_proto985 OR …)^10

The coordinates of each key point are transformed into polar coordinates (t,r) with respect to the center of the image. These coordinates are used during search to boost matching key points that have similar polar coordinates (i.e. key points that are in the same portion of the image). For example a query key point represented as proto385 matches all 'proto385' key points but the score that this match contributes to the overall score of the image is boosted based on the distance (D) between the query key point and the hit key point. The boost factor is inverted with respect to D, i.e. the larger the distance the lower the boost. This means that matching key points that are very close to the query key point get higher scores.



Relevance score

The default Lucene [8] similarity strategy was used to calculate relevance scores for documents that match the given query. In essence the similarity is based on TF.IDF with the addition that the score resulting from a query term can be changed using boosting or based on payloads.

$$docscore(d|Q) = \sum_{t \in Q} \left(tf(t \in d) \cdot idf(t)^2 \cdot boost(t \in d) \right)$$

Docscore(d|Q) = score for document d given query Q
t = term from query Q
tf(t∈d) = term frequency of t in document d
idf(t) = inverse document frequency of t
boost(t∈d) = boost factor based on t in d

The boost factor is based on the coordinates of the query term (i.e. a single key point) with respect to the coordinates of the key point that was matched in the document as described before.

In the end the task is to find relevant clips based on one or more query images. This is done by searching on individual query images and aggregating the result. Each possible clip is fetched and initiated with a score of 0.0.

A search for a query image results in a list of matching images ordered on their relevance score. For each result image its clip is fetched its score is incremented with the relevance calculated by the search engine. Finally the list with clips is ordered based on their score and returned as the final result.

Submitted runs

We submitted two runs using this method: F_X_NO_TNO-INOMSK_2 and F_X_NO_TNO-INORDR_4 which only differ in the use of masks or not. For both methods we used the following query model:

- Use of 'Best_proto' field with a boost factor of 1 (focus on recall) in combination with
- 'Prototypes' field with a boost factor of 10 (focus on precision).

- Polar coordinates were used.

The INORDR method used all key points found in a query image while the INOMSK method only used key points within the mask for search.

Observations

The idea behind the representations of key points and the use of wildcards works but has proven to be too slow to be of actual use. For example a query like: *proto385-0.03???_proto041-0.00???_proto795-0.00???* results in less results than: *proto385-0.0????_proto041-0.0????_proto795-0.0????*. In addition the results of the first query are usually better but tend to miss relevant results as well (its more tuned for precision rather than recall). Unfortunately the use of wildcards is very time consuming since the search engine first has to find all possible terms that match the wildcards, expand the query and perform the search. Doing this for some hundred to multiple thousand terms for a single query image was not feasible.

The representation used (i.e. proto385_proto041_proto795) either with or without scores resulted in a very large number of unique words (almost a billion possible words). This poses a problem for the search engine that uses an inverted index, pointers from words to documents, for fast retrieval. Having this many words (a normal language has a maximum of 1 million words) results in a very large inverted index which makes searching relatively slow (or very memory consuming).

Results

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

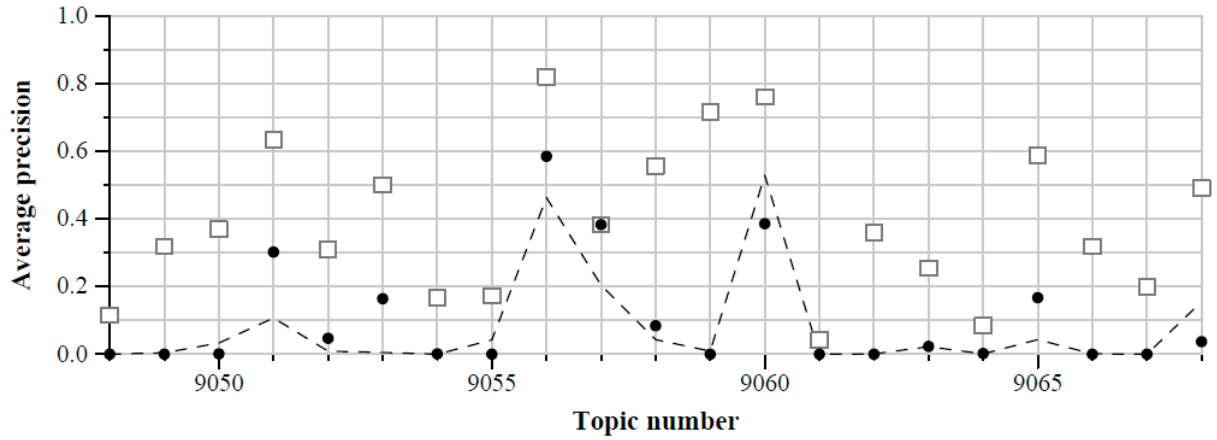


Figure 4: Average precision of run F_X_NO_TNO-ANNMSK_1 (dot) versus median (---) versus best (box) by topic.

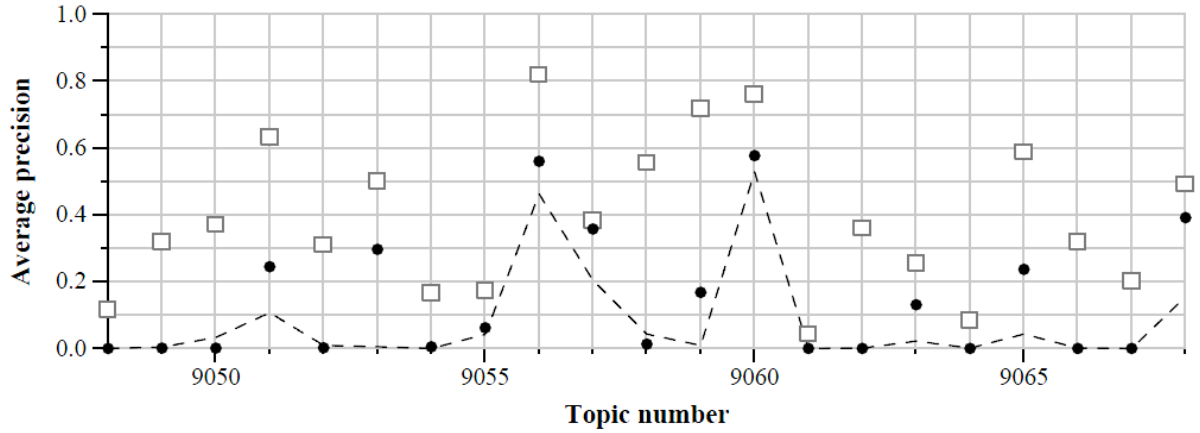


Figure 5: Average precision of run F_X_NO_TNO-ANNMSK_3 (dot) versus median (---) versus best (box) by topic.

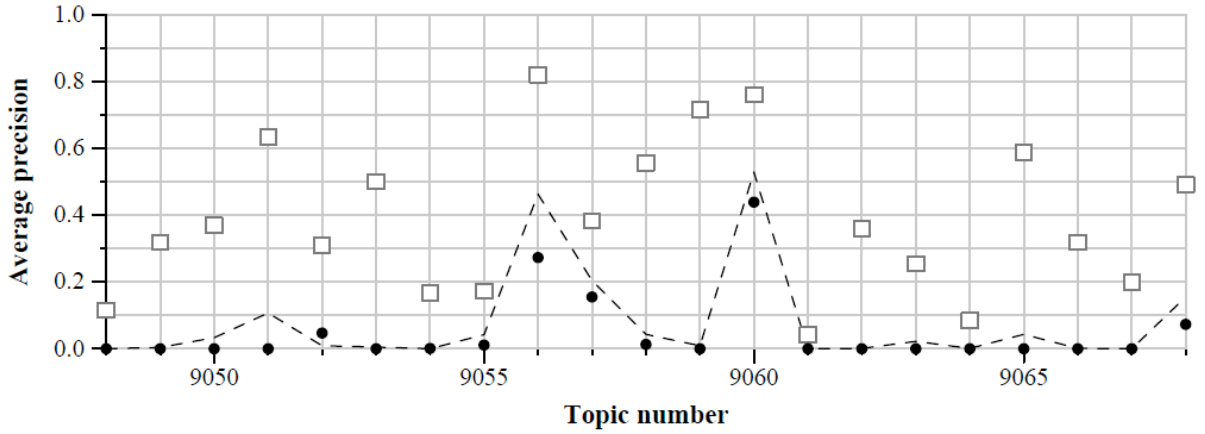


Figure 6: Average precision of run F_X_NO_TNO-INOMSK_2 (dot) versus median (---) versus best (box) by topic.

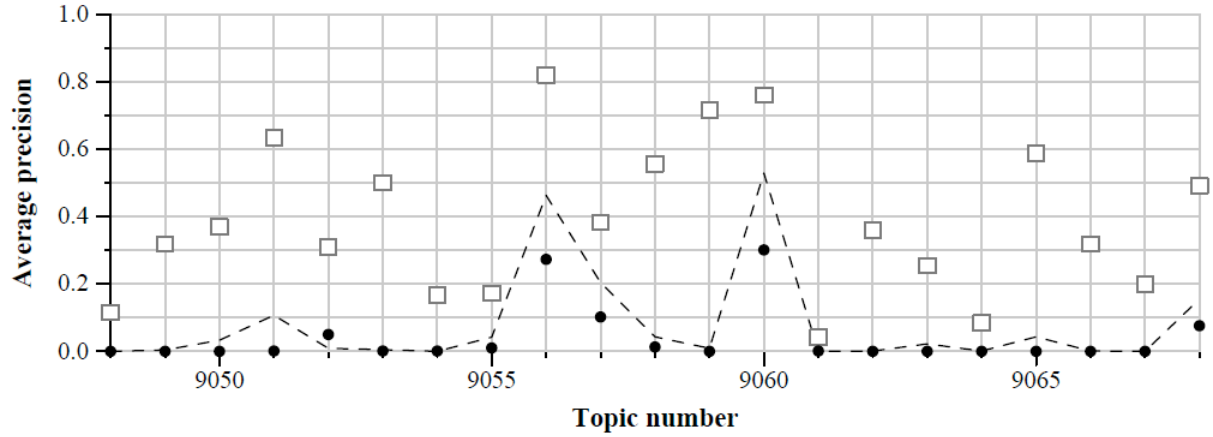


Figure 7: Average precision of run F_X_NO_TNO-INORDR_4 (dot) versus median (---) versus best (box) by topic.

From these figures it is clear that TNO run 1 and 2 perform best w.r.t. the other two runs and average performance, for one topic (9057) run 1 was the best performing one. TNO run 3 & 4 (bag-of-words) have average results.

Conclusions

(F_X_NO_TNO-ANN*) Using a brute-force search over all SIFT key point descriptors gives real instance search results: hits for every query. However, there is no generalization of objects and it takes relatively long (15-30 minutes on a single-core PC for a query set). Furthermore, when the object is small (for example query 9048) and there are not many SIFT key points inside the query mask the search becomes difficult.

Figure 8: Query 9059 is the ideal query: nine example images, no masking: use the whole image and as a result many key points to search with.

(F_X_NO_TNO-INO*) Our extended bag-of-words approach gives for some queries nice concept detection results and is fast but does not perform as expected for instance search.

Acknowledgements

Part of the work described in this paper was supported by Dutch project ‘Metadata Extraction Service’ (MES), within the Dutch government ‘Service Innovation & ICT’ programme.

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] http://lucene.apache.org/core/