Dissimilarity Detection of Two Video Sequences

Technical Report - FIT - G20102015006 - 2012 - 04

Vítězslav Beran
Lukáš Klicnar

Faculty of Information Technology, Brno University of Technology

December 9, 2012

Abstract

The technical report presents the evaluation of video processing methods focused on visual content description in the scope of the VideoTerror project. The video processing task for detection of short- and long-term changes between two video sequences is defined in detail. The algorithm comparing two video sequences (reference and query) is introduced together with definition of particular situations that the algorithm must be able to detect: re-written parts, removals or injected parts. The image processing methods are selected to be robust to several practical distortions that might appear in defined task. The appropriate computer-vision methods are presented and discussed, then proposed method and experiments are introduced and evaluated on manually generated dataset proposed for VideoTerror particular tasks. The algorithms are selected and optimized to be effectively integrated into VideoTerror hardware solution. The technical report finally describes the algorithms developed tool, its usage, parameters and output formats.

Contents

1 Introduction

The main task of the video comparison systems is to detect and validate the differences between two visually *almost* identical video sequences. In some cases, even when two video sequences are declared as the identical, small differences might appear and manual detection and validation of such video parts may become extremely time consuming and unbearable. The example of video-pair disruption with dissimilarity types is showed on Figure 1.

Figure 1: Examples of video-pair disruption.

When comparing the similarity between two video sequences (reference and query), we define three types of dissimilarity that might occur:

- rewriting - part of the query video is rewriten by different visual content than in reference video and the length of query video part is the same as the reference video part;
- injection - part of the query video is new - added to original (reference) content, so the query video part is longer than the reference video part;
- removal - part of the query video is removed, so the query video part is shorter than the reference video part;

Presented research is focused on visual content, so audio is omitted. The visual part of the video is sequence of consecutive images, video frames, and one way of evaluating similarity between two videos (or its parts) is to compare the similarities between video frames and compute statistical analysis.

In out work, we represent the video sequence as the set of video-parts. Each video-part is represented by its temporal information (begin and end) and also by one or more key-frames. The video-part key-frames are in some sense interesting video frames and are usually represented by image descriptors. One of the research goals is to analyse the influence of the density of the video-part key-frames to stability and precision of the entire video-pair comparison approach.

First, the overview of the state-of-the-art methods together with some optimizations and novel approaches is presented in Section 2. The developed algorithm is proposed and its crucial parts (key/frame extraction, similarity matrix, geometrical validation and dissimilarity detection) are discussed in Section 3. Results of experiments with selected video processing algorithms and

developed methods are presented and discussed in Section 4 together with utilized datasets, both widely used in video processing community and manually created for particular VideoTeror tasks. Finally, the tool is realized and its usage, parameters and output formats are described in details in Section 5.

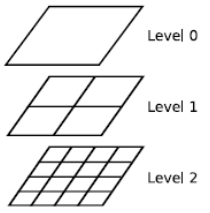
2 Video data analysis and representation

The similarity between two images (video frames) can be in general evaluated using two types of visual content description: *global* and *local*. The *global* approach of image description extracts image features from the entire image and utilizes statistics for their representation. Global approach might be very computation cost effective but is usually not very robust to geometrical distortions as no spatial information is taken into account.

2.1 Global image features

We represents the image content by colour histograms combined with a spatial pyramid over the image to jointly encode global and local information [3]. We use several colour models (grey-scale, HSV, IO_1O_2). The IO_1O_2 colour model, known as the opponent colour model [4], is partially colour normalized and simple to compute. The spatial pyramid is arranged so that low number of bytes of data is describing each pyramid level. These are appended to create the final feature vector. On descending to the next level in the pyramid, the number of segments the histograms are taken over increases four-fold. Therefore, to maintain the size constraints for each level, the number of bytes used to describe each channel per segment is quartered. This places a desirable bias on the importance of the levels. The representation is illustrated on Figure 2.

The amount of data for storing the intensity channel is two times bigger than the other opponent colour channels. This is a common practice as generally more information is contained in the intensity information. The histograms are then L1 normalized. The Euclidean distance between image feature vectors is a meaningful measure of similarity.



Level	I	O_1	O_2
0	64	32	32
1	16	8	8
2	4	2	2

Figure 2: Spatial subdivision of the image at each level of the histogram pyramid and the amount of data stored for each segment [3].

Besides colour information, we compute also histograms of image gradients to represent the image intensity changes.

2.2 Local image features

The *local* approach extracts local image features such as corners or blobs and represents the image content as the set of such local features and their descriptors (see Figure 3 for example of local image features represented by yellow circles). The local approach is more robust to geometrical distortions but might have poor results with noisy data and is computationally more expensive.

Figure 3: Examples of detected scale-invariant local image features.

The methods based on *Harris corner* detector [10] and *Hessian matrix* [11] proposed by Mikolajczyk and Schmid introduces principles of corner and blob image structures detection that are invariant to structure scale. One of the widely used approach is the SIFT detector proposed by Lowe [8] for its high spatial and scale precision of detected local features and also because it includes also very robust method for feature description. Next favourite method that accelerates the Hessian-based approach is known as SURF detector [1]. The SURF method is also robust to scale and is computationally very effective because is based on integral-image representation and further approximates the second-order derivative computation by box-filters. The approach introduced by Rosten and Drummond known as FAST corners [12] employs machine learning to construct corner detector that outperforms all know approaches in the speed point of view. The FAST corner method is not so precise and stable as SIFT or SURF method, but for several applications give sufficient results with extremely low computational cost. Other region-based detector with promising performance is the MSER (Maximally Stable Extremal Regions) developed by

Matas et al. [9]. It detects image regions that all pixels inside the region have either higher or lower pixel intensity than all the pixels on its outer boundary. The method can be efficiently implemented and attains good robustness and repeatability. Several other scale-invariant interest point detectors have been proposed. Examples are the salient region detector proposed by Kadir and Brady [6], which maximises the entropy within the region. From the subset of methods based on edges and edge regions the *edge-based region* detector proposed by Jurie et al. [5] or detector by Tuytelaars and Van Gool [16] have the interesting performance. They seem less amenable to acceleration though.

2.3 Bag-of-words and visual codebooks

When images are represented as the sets of descriptors, the time complexity of the comparison of two images is not insignificant as each descriptor from one image must be compared to all descriptors from the other image. In tasks based on image comparison such time complexity become unbearable. Representation of descriptors as single terms defined by some codebook may rapidly change the computational cost of the comparison operation.

The idea of visual codebook introduces the techniques from natural language processing and information retrieval area into computer vision [15] and is mostly called as *bag-of-words*. The approach is based on the Vector Space Model [17] that computes a measure of similarity by defining a vector representing each document. The model is based on the idea that the meaning of a document (image) is conveyed by the used words (image structures). The two image content is then compared using cosine distance of two image bag-of-words. The bag-of-words construction is showed on Figure 4 and also weights are presented based on inverse document frequency. More detailed description of codebook training methods and bag-of-words construction was published in [2]. Bag-of-words approach is very efficient for fast image comparison and retrieval (for inverted file index construction), but keeps the high computational cost drawback of local feature extraction approach.

2.4 Temporal analysis

The temporal analysis is usually used in video processing to analyse the geometrical changes in consecutive video frames. According to application, the analysis serves e.g. to detect cuts in video sequences or find the visually most representative candidates of video-parts. We have selected two distinct approaches. First approach computes the differences between several adjacent video frames represented by global image features using Euclidean distance for metric features and cosine distance for bag-of-words representation. The differences are evaluated over flowing window. Figure 5 shows the output signal with candidates video-part boundary candidates (red lines) and key-frame candidates (blue lines).

Other approach is based on tracking of local image features over the close video frames. The approach is motivated by work of Sivic et al. [14] and further

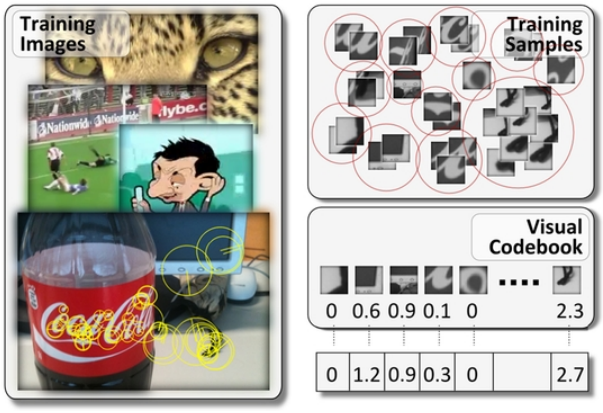


Figure 4: Bag-of-words construction overview - local image features are translated by codebook and weighted.

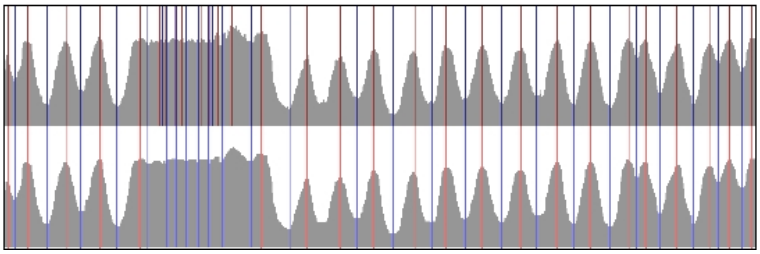


Figure 5: Similarity of visual content of adjacent video frames with shot boundary (red) and key-frame (blue line) detected candidates.

developed by Klicnar and Beran [7] for computationally efficient video segmentation. The existing method was adapted to a higher computational speed and on-line processing. The proposed approach is based on sparse local image features and the KLT tracker for feature trajectory computation. A RANSAC-based method is used for initial motion segmentation, resulting motion groups are partitioned by a spatial-proximity constraints. The correspondence of motion groups across frames is solved by one-frame label propagation in forward and backward directions. The method results in stable trajectory bundles that represents distinctive image regions. The Figure 6 shows the steps of the procedure (the tessellation is applied arbitrarily).

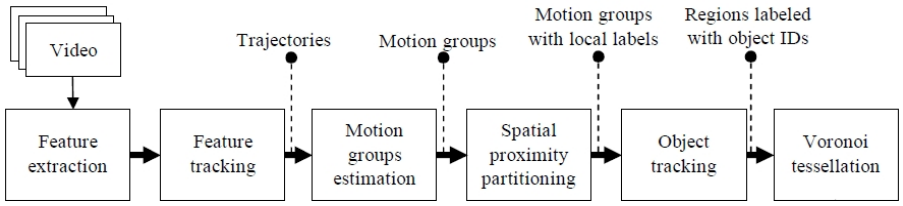


Figure 6: Similarity of visual content of adjacent video frames with shot boundary (red) and key-frame (blue line) detected candidates.

3 Video sequence visual-based comparison

This section describes basic principles of the proposed system for video dissimilarity detection. Its design consists of several independent consecutive layers (see also the block diagram on Figure 7):

1. Preprocessing of both, reference and query video sequences
2. Computation of similarity matrix
3. Detection of corresponding segments in both videos

Video sequence preprocessing actually involves two important steps. The system doesn't work with all frames of the sequence, this would be very time-consuming and it wouldn't improve the resuluts significantly. Instead of this, only a set of keyframes is used – their extraction makes the first step of preprocessing. An image descriptor is computed for every keyframe, which is the way these frames are represented in the system. Comparison of two frames is then reduced to computation of distance of their descriptors and the main advantage is that the descriptors can be designed to be robust to some image distortions and transformations. Finally, the output of this layer is the representation of the whole sequence by a set of keyframe descriptors. It is independent of the input sequence and it can be stored e.g. in the database.

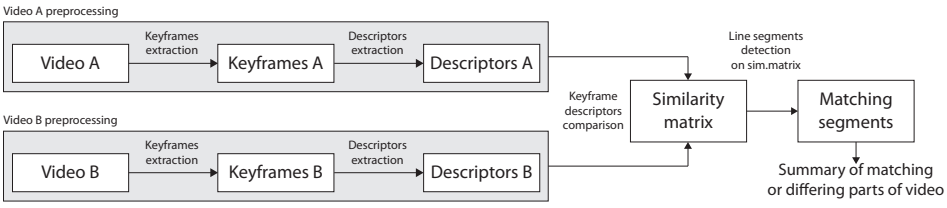


Figure 7: Basic block diagram of the proposed system for video sequence comparison.

Set of keyframe descriptors for both sequences is the input of the next layer, which is called similarity matrix computation. In this step, all keyframes from one sequence are compared to keyframes from the other one and this results to a matrix, which can be understood as a map of similarity of both sequences. This matrix is thresholded, preprocessed to remove noise and it is the only input of the next step, detection of corresponding parts in both videos. This task consists of detection of continuous line segments in the matrix, which can be interrupted and shifted. It is handled by a recursive algorithm we developed. Finally, if we have information about matching parts in both sequences, the decision about changes is straightforward, they correspond to discontinuities in detected line segments..

3.1 Key-frame extraction

The goal of the keyframe extraction is to describe to whole video sequence by a set of keyframes, which represents the individual video parts. Every part (or it can be called a segment) is described by several keyframes - the start frame, the end frame and possibly one or more representative frames inside. Having only one representative keyframe is suitable when all frames inside the segment are fairly similar, but even in this case, our experiments showed that when using our similarity matrix-based approach for video comparison, it is better to have more inner keyframes. If a new keyframe is periodically, dense enough created inside an every segment, it forms a stronger line-response on the matrix, which is more easy to detect, as it is far more distinct from the noise patterns.

Actually, it is possible not to detect segment boundaries, but create keyframes directly by using every Nth frame of the video sequence. This may work fine in some cases, but the precision of detection of their boundaries is dependent on the period N and very short segments cannot be detected – they may be simply passed unnoticed if they lie between two keyframes. This is the reason why every frame must be inspected and boundaries of the individual video parts must be found. When all keyframes are extracted, they are described by frame descriptors.

The block diagram of the proposed keyframe extraction method is presented on Figure 8. Frames are processed sequentially and we check for discontinuities in the video. A significant change in the video is considered as a boundary between two segments, so detection of a such change directly leads to a keyframe creation. In addition, keyframes are also create every 25 frames (resp. 1 second at common framerate of 25 fps), which improves the response and stability of long segments in their detection process. So, the resulting set approximately contains a keyframe every 1 second combined with keyframes at segment boundaries.

The last thing is that how segment boundaries are detected, resp. how we determine the significant change that is referred in Figure 8. We tried two

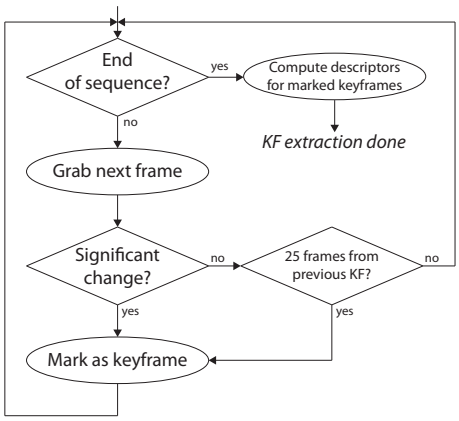


Figure 8: Block diagram of the keyframes extraction process.

approaches - by observing the global features or by utilizing the local features and their development in time.

3.1.1 Local features

Our solution is based on [7], specifically we utilized the tracking part and used it for keyframes extraction. We assume that every segment represents a comprehensive part of the video sequence, it can change, but these changes are gradual and most importantly, continuous. We track a set of interest points in the image – some of them disappear, new ones are detected, but most of them can be tracked stably over a longer base of frames. If significant amount of trajectories breaks at one time, it will be considered as a big change, start of new segments. Criterion is the following:

$$\frac{n_{broken}}{n_{total}} > N \tag{1}$$

where n_{total} is the number of all tracks, n_{broken} represents count of currently broken tracks, N is a threshold. Because we actually track objects (although we don't recognize point of these objects as a one unit), we can consider this as a content-based keyframes extraction.

3.1.2 Global features

The existing method representing the image content by colour histograms combined with a spatial pyramid over the image was presented in [3]. We applied the pyramidal approach also for histogram of image gradients. Further novelty of our method is interpolation of segment histogram values into adjacent bins to improve the descriptor robustness to small geometrical distortions. Similar to

SIFT descriptor, the Gaussian weighting function is applied to segment image values (intensities or gradient lengths).

3.2 Similarity matrix

Similarity of the extracted keyframes is described by a so called similarity matrix S. Its rows represent keyframes from a reference sequence V_R, while columns represent keyframes from a tested sequence V_T. Every value in this matrix represents the dissimilarity of keyframes $V_R(r)$ and $V_T(t)$, in our case it is the distance between descriptors of both keyframes $S(r,t) = d(V_R(r), V_T(t))$. This matrix can be obtained by comparison of every combination of keyframes, which is the way we use. Very useful step is thresholding by a threshold T, new similarity matrix S_T is created:

$$S_T(r,t) = \begin{cases} 1 & \text{if } S(r,t) < T \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Threshold T is the maximal distance of descriptors for two frames that are considered similar. Example of several thresholded similarity matrices is on Figure 9. As can be seen, similar parts of both video sequences forms evident diagonal line segments in the matrix. In case of comparison of the same sequences, strong line on the main diagonal emerges. If the video contains scenes with very slow changes (so many subsequent frames are very similar), squared structures around the diagonal lines are created. When comparing sequences with no similarities, no diagonal lines appear in the matrix.

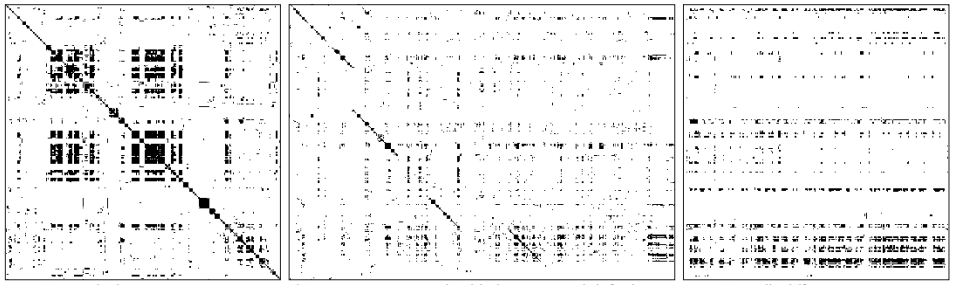


Figure 9: Examples of thresholded similarity matrix for various situations – from exactly the same video sequences to completely different ones. Black color represents values above threshold (high similarity of frames), while white color represents values under threshold (low similarity).

Similarity matrix obtained by comparison of every keyframe combination usually contains significant amount of noise caused by random frames similarity, which can form small structures or segments in the thresholded matrix. These

segments are fortunately usually different from diagonal lines that are the only important parts of the matrix. That's why we filter the unwanted segments by convolving the matrix with the following kernel K (which has strong response for diagonal lines) and thresholding again:

$$K = \frac{1}{5}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

3.3 Dissimilarities detection

As the corresponding parts of both video sequences appears as strong line segments (with high frame-to-frame similarity) on the similarity matrix, the video matching problem can be reduced to searching for these lines. In the proposed algorithm for this task, we suppose several assumptions: First, the frame rates of both video sequences doesn't differ excessively, so the lines are nearly or exactly diagonal. We also assume that the order of the scenes is preserved, only some of them are removed, replaced, or there is some other content inserted. This means that the disconnected line segments on the similarity matrix can only be shifted to the right and/or downwards from the previous segment, which is utilized in the way that the matrix is processed.

We developed an recursive algorithm for segments detection based on the *divide and conquer* technique. First, we need to define a continuous segment, which is a line of neighbouring points on thresholded similarity matrix S_T that goes from $A = (a_x, a_y)$ to $B = (b_x, b_y)$. From a given starting point $A_1 = (a_x^1, a_y^1)$, the segment can be gradually constructed by following the diagonal or by doing a vertical/horizontal step:

$$A_{n+1} = \begin{cases} (a_x^n + 1, a_y^n + 1) & \text{if } S_T(a_x^n + 1, a_y^n + 1) = 1 \\ (a_x^n + 1, a_y^n) & \text{if } S_T(a_x^n + 1, a_y^n + 1) \neq 1 \wedge S_T(a_x^n + 1, a_y^n) = 1 \\ (a_x^n, a_y^n + 1) & \text{if } S_T(a_x^n + 1, a_y^n + 1) \neq 1 \wedge S_T(a_x^n + 1, a_y^n) \neq 1 \wedge S_T(a_x^n, a_y^n + 1) = 1 \\ (a_x^n, a_y^n) & \text{otherwise.} \end{cases} \tag{4}$$

where T is the threshold for similarity matrix values. The segment is constructed until $A_{n+1} = A_n$. The length of the segment is defined as Euclidean distance: $d(A, B) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$.

Basic block diagram of the proposed approach is on Figure 10. At first, the whole matrix is searched for the point, from which the longest continuous segment can be constructed. If the length of this segment $d(A, B) > d_{min}$, it is accepted and the matrix is subsequently divided into these three areas (as illustrated in fig. 11):

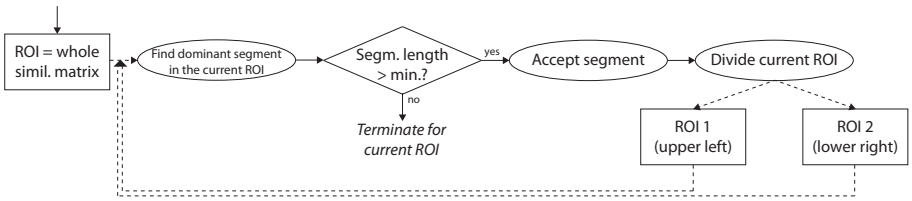


Figure 10: Block diagram of the similarity matrix processing.

1. ROI 1: Rectangle from upper left corner to start of the segment
2. ROI 2: Rectangle from end of the segment to lower right corner
3. Remaining areas

ROIs (regions of interest) 1 and 2 are then processed recursively. In each of them, the dominant continuous segment is detected and if fulfils the minimal length criterion, it is accepted, the region is subdivided and the recursion is repeated. Optionally, an angle criterion can be involved, which ensures that the detected segment is nearly or exactly diagonal. We suppose that longer continuous segments are less probable to be formed by noise, so the extraction of the most dominant lines as first improves the robustness of this algorithm.

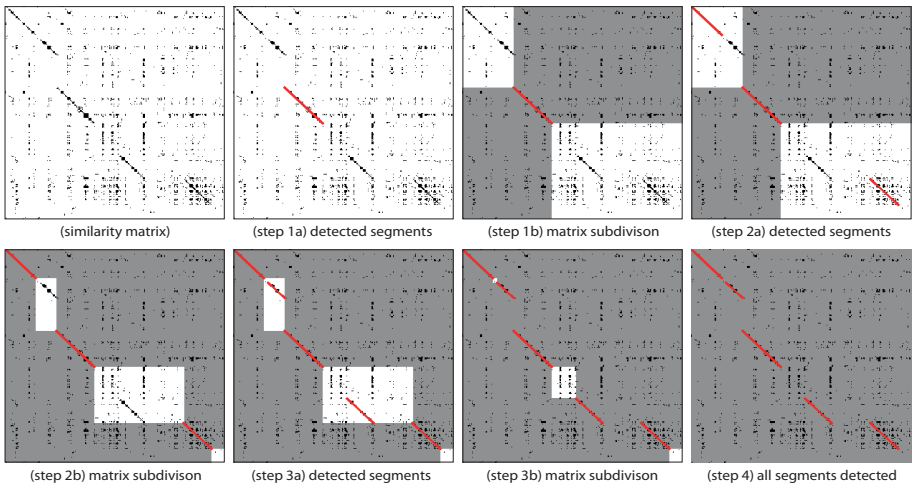


Figure 11: Example of dominant segments detection.

The Figure 11 shows how this algorithm searches for the new line segments in the rectangular areas between the already found ones. Also, if both video

sequences are exactly the same, only the diagonal segment is detected and the algorithm terminates after one step. It is also obvious that the diagonal is the longest possible line, so it will be always detected as the first one.

The left picture shows the thresholded similarity matrix, which is used for demonstration. The white colour represents areas of the matrix for the next segment detection, while the grey colour marks already inspected or rejected regions, where no further segments can be detected. At the beginning, the largest segment is extracted (step 1a) and the matrix is subdivided into two parts (step 1b), where the same procedure is proceeded recursively (steps 2,3). At the end, only too short segments can be detected in the remaining areas, so the recursion is terminated and five segments are detected.

4 Results

This section consists of four main parts. The first one describes datasets used for experiments, it contains brief characteristics of included sequences and it is divided into two groups – for evaluation of frame descriptor behavior and for evaluation of matching segments detection peformance. Types of artificially added distortions are also described. The second part of this sections deals with experiments with descriptors, it is mainly a free quotation of [13]. In the third part of this section, we evaluate performance of the video comparison as a whole unit. Experiment details and used metrics are introduced as first, the results are then shortly discussed. The last, fourth, part addresses the question of computational demands.

4.1 Datasets used for experiments

We used a small dataset for experiments and evaluation. Because of the necessity of different properties of the data for evaluation of diffent parts of the video matcher system, we divided the data into two basic groups: The first one is for experiments of descriptor performance and the second one serves for evaluation of the video comparison itself.

4.1.1 Data for descriptor performance tests

Descriptor performance tests are taken from [13], so only brief description of the dataset the author used follows. Nine different video sequences were use to represent varying types of scenes – sports, news, cartoons, etc. Every video is strongly specific, some of them contain fast cuts or static scenes, gradually moving camera, large color variability, etc. Every sequence has a resolution of 640x360px and a framerate of 25 fps. All videos were downloaded from YouTube, specifics of the sequences are summarized in the following description:

- **vid01** – Climbing a table-top mountain - Expedition Guyana - BBC

  - Recording of climbing ascent. Shots of landscape, rocks and details od mountain climber. Mostly slow camera movement, sometimes even static, cuts to very different shots.
  - 5977 frames

- **vid02** – Annapurna Base Camp Trekking
  - Expedition to Annapurna. Shots of people groups, villages, landscape, river, mountains, sometimes inserted static images.
  - 30618 frames

- **vid03** – Chile Extreme Whitewater Kayaking
  - Short movie about kayakers at wild water. Shots of river, watefalls, detailed shots of kayakers., fast cuts to very similar scenes
  - 8661 frames

- **vid04** – Wild Chronicles: Madagascar Poison Frogs
  - National Geographic document about poison forgs. Shots of rain forests and animals living in them, cuts from detailed shots of animals to landscapes, etc.
  - 9111 frames

- **vid05** – CNN BREAKING NEWS - NASA SOLAR STORM WARNING
  - Short passage from CNN news. Mostly shots of anchors with illustrative graphics, relatively static scenes.
  - 3048 frames

- **vid06** – 24H Nurburgring 2011 START
  - Part of car race recording. Mostly shots of cars, fast camera movement, in some places shaking, very similar shots.
  - 3264 frames

- **vid07** – CORAL REEF
  - Documentary about coral reefs. Underwater scenes, very similar, large surface with the same color, subtitles.
  - 8638 frames

- **vid08** – The Hobbit Trailer
  - Movie trailer. Mostly shots of people in different situations, very fast cuts to scenes of similar colors, included information graphics, subtitles.
  - 3780 frames

- **vid09** – Donald Duck Put-Put Troubles
  - Short cartoon movie. Quickly changing scenes, but often just part of the frame, very similar large surfaces, scenes of similar colors.
  - 10758 frames

These video sequences were modified for testing of different types of distortion. These were identified as the main three groups: The first one is **resize+crop**. Every video was enlarged to 110%, 120%, 130% of its original size and cropped, which results to loss of data at the borders. Illustration is on Figure 12.

Figure 12: Example of the resize+crop distortion.

The second type of distortion is the **change of quality**. This was simulated by increasing brightness to 105%, 110%, 115% of original, by lowering contrast to 85%, 70% 60% of original and by Gaussian-bluring with kernel of size 2, 4, 6px. Illustration is on Figure 13.

The last distortion is the **compression**. Modified sequences were obtained by compressing the original with the DivX encoder with different bitrates. The compression significantly damages the details of the image. Illustration is on Figure 14.

4.1.2 Data for video matching tests

The main specifics of the data to video matching evaluation is that we need actually two sequences – one is the original, we call it **reference**, which is the

Figure 13: Example of the quality distortion.

sequence without modifications. The second one is called **query**, it may contain some edits as described earlier and we want to find the parts that are equal in both sequences. We created a small dataset that we used for algorithm design and its evaluation. It consists of a few artifical situations – we took a video sequence and edited it, some content was excluded or replaced, some new parts were added. Together, we used:

- **music** – Music videos (rewriting detection)
  - Two music videos of Metallica and Apocalyptica bands, contains scenes with static or slow moving camera, colors of both sequences are very different.
  - Reference sequence was made by replacing parts of one sequence with parts of the second one, so as a query sequence, we can use each of the basic ones.
  - Reference – 5582, query A – 5079, query B – 9678 frames.
- **tv1** – TV series 1 (removal detection)
  - Five minutes cut-out from a episode of the Big Bang Theory series. Contains similar scenes, slow moving camera.
  - Reference is the whole five minute sequence, query was shortened by one minute from the start and from the end. It is made for testing of shortening detection.
  - Reference – 7506, query – 4488 frames.

Figure 14: Example of the compression distortion.

- **tv2** – TV series 2 (all dissimilarities)
  - Five minutes cut-out from the same episode of the Big Bang Theory series, which is the reference sequence. One part was replaced by a 30s from another TV series (How I Met Your Mother), another 30s part was excluded, another 30s part was inserted.
  - Reference – 7503, query – 7503 frames.
- **tv3** – Movie (all dissimilarities)
  - Same as the tv2, except that 5min part of the movie Groundhog day was used as a reference sequence.
  - Reference – 7501, query – 7501 frames.

The music sequence is downloaded from YouTube and has a resolution of 384x288, the other ones are grabbed from a digital TV broadcasting, edited and resampled to the same resolution.

4.2 Descriptor performance

The goal was to determine, how different distortions affect the frame descriptors distances. Following description and results are taken from [13], where the author used similarity matrix for reference and distorted sequence comparison. First, similarity matrix was computed for every pair of reference and query sequence, only every 25th frame was used. Example of such matrix is on Figure 15. Black colour means that frames are identical (low, resp. zero descriptor

distance), white colour means that frames are completely different (large distance) – the darker the colour is, the more similar the frames are. Because the content of both sequences is similar, it is obvious that the diagonal represents comparison of corresponding frames. A strong diagonal line can be clearly seen on the left-most figure, but it becomes weaker with more distortion that the descriptor is not tolerant to.

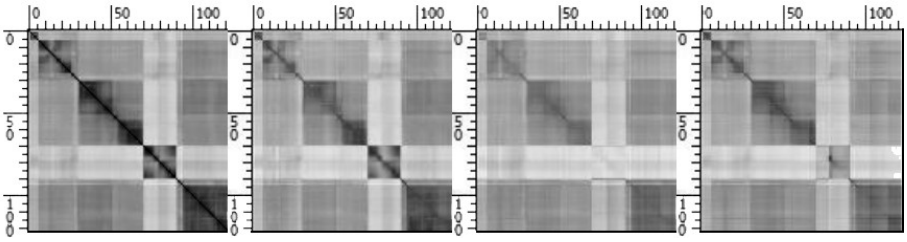


Figure 15: Similarity matrices for **vid05**, using the colour histogram descriptor. The reference sequence was compared to (from left): reference, resize+crop, quality, compression, all modifications represents the most distorted variant (level 3).

These matrices were made for every sequence from the dataset described in Section 4.1.1, every sequence was compared to all of its distorted versions (see example on Figure 16). Next, every similarity matrix for distorted sequence was compared according to the following metric, so the error was computed:

$$E(S_{ref}, S_{query}) = \frac{\sum_{y=1}^{Y} \sum_{x=1}^{X} |S_{ref}(x,y) - S_{query}(x,y)|}{XY}, \tag{5}$$

where S_{ref} (resp. S_{query}) are the similarity matrices created by comparison of the reference sequence with the reference (resp. query) sequence. Sizes of both matrices are the same, as the compared sequences are equal (except for image distortion).

Example of results for the colour histogram image descriptor are shown on Figure 16. It is obvious that change of quality and compression significantly affects the descriptor distance. The descriptor showed the best tolerance for resize+crop, but this is only because the crop was very small and Gaussian-weighting of the tiles simply suppressed the change. But from our experience comes that behaviour of this descriptor is hugely affected by cropping the image. The gradient variant from [13] handles better the change of quality, because it preserves the image structure, but it is more prone to compression, which on the contrary destroys the structure.

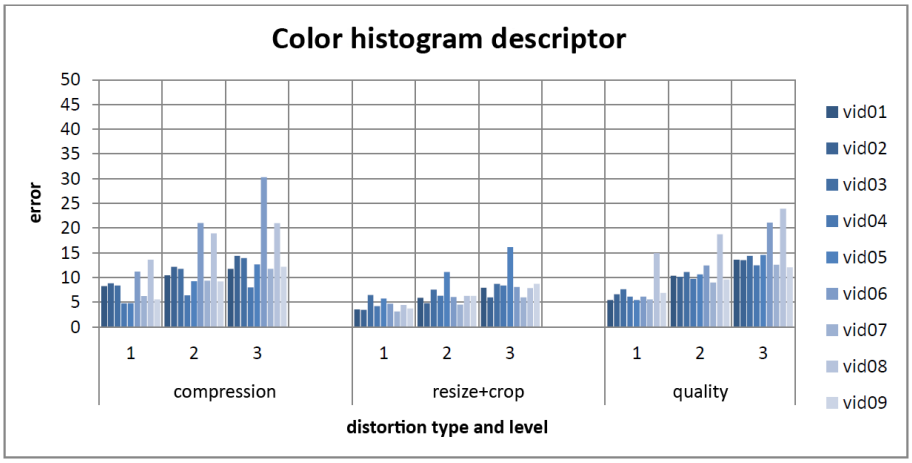


Figure 16: Performance of the colour histogram descriptor.

4.3 Video matching performance

The aim of these test were to evaluate, how the similar segments extraction performs in different situation. First, we computed a similarity matrix by comparing a reference sequence with the one with added disruptions. The task is to find the equal parts present in both sequences – they arise as line segments on the similarity matrix. We used the video sequences described in Section 4.1.2 and we pursued these following standard evaluation metrics:

- **Segments count** – Number of segments that are actually present on the similarity matrix.
- **Segments found** – Number of segments that was detected on the similarity matrix.
- **Recall** – The fraction of segments that are detected, which is considered true if at least one detected segment corresponds with the actual one. It is computed by the following formula:

$$\mathrm{R} = \frac{\#\text{ of present seg. that are detected}}{\#\text{ of all present seg.}} \tag{6}$$

- **Precision** – The fraction of all found segments that are correct, which means that the detected segment corresponds to an actual one. It is computed by the following formula:

$$\mathrm{P} = \frac{\#\text{ of detected seg. that are correct}}{\#\text{ of all detected seg. (overseg. count as 1)}} \tag{7}$$

- **Over-segmentation** – Penalizes situations, where some segments are actually detected as a multiple shorter ones. It is computed by the following formula:

$$\mathrm{OS} = \frac{\text{for each pres. seg.:} \sum \#\text{ of corresp. that are detected}}{\#\text{ of correctly detected segm. (overseg. count as 1)}} - 1 \qquad (8)$$

The results are summarized in Table 1 and the corresponding thresholded similarity matrices with highlighted actual found segments are shown on Figure 17. Special case is the comparison #1, which compares a sequence with itself. Some of the matrices are very noisy, which is caused by very similar frames across the whole sequence, but the detection algorithm works very well – all of the present segments were found in all evaluated sequences. But several imperfections are present, they could be generalized as:

- **False segments detection** – some of the noise patterns can be detected as segment by mistake, for example in comparison #1 (tv2).
- **Inaccurate localization** – as can be seen in comparison #1 (tv1): The detected line doesn't exactly lie on the actual one. In this case, it is caused by the way the line segments are gradually constructed – horizontal or vertical steps are not limited, so it continues over a black rectangle of very similar frames and shifts one end of the line.
- **Over-segmentation** – one segment is detected as multiple ones, because a line on the similarity matrix is interrupted. This may be caused by bad tollerance of the descriptor to some distortion, that is present in one or several frames.

#	**Reference**	**Query**	**# seg.**	**# found**	**R**	**P**	**OS**
1	music qr. A	music qr. A	1	1	1.0	1.0	0.0
2	music ref.	music qr. A	3	3	1.0	1.0	0.0
3	music ref.	music qr. B	4	5	1.0	1.0	0.25
4	tv1 ref.	tv1 qr.	1	1	1.0	1.0	0.0
5	tv2 ref.	tv2 qr.	4	5	1.0	0.8	0.0
6	tv3 ref.	tv3 qr.	4	4	1.0	1.0	0.0

Table 1: Results of the matching segments extraction for different video sequences.

The protocol for evaluation of video-matching method performance is designed to be scalable for bigger datasets (TRECVID) for other experiments. Actually, the size of the used dataset (designed to reflects the particular task) biases the performance.

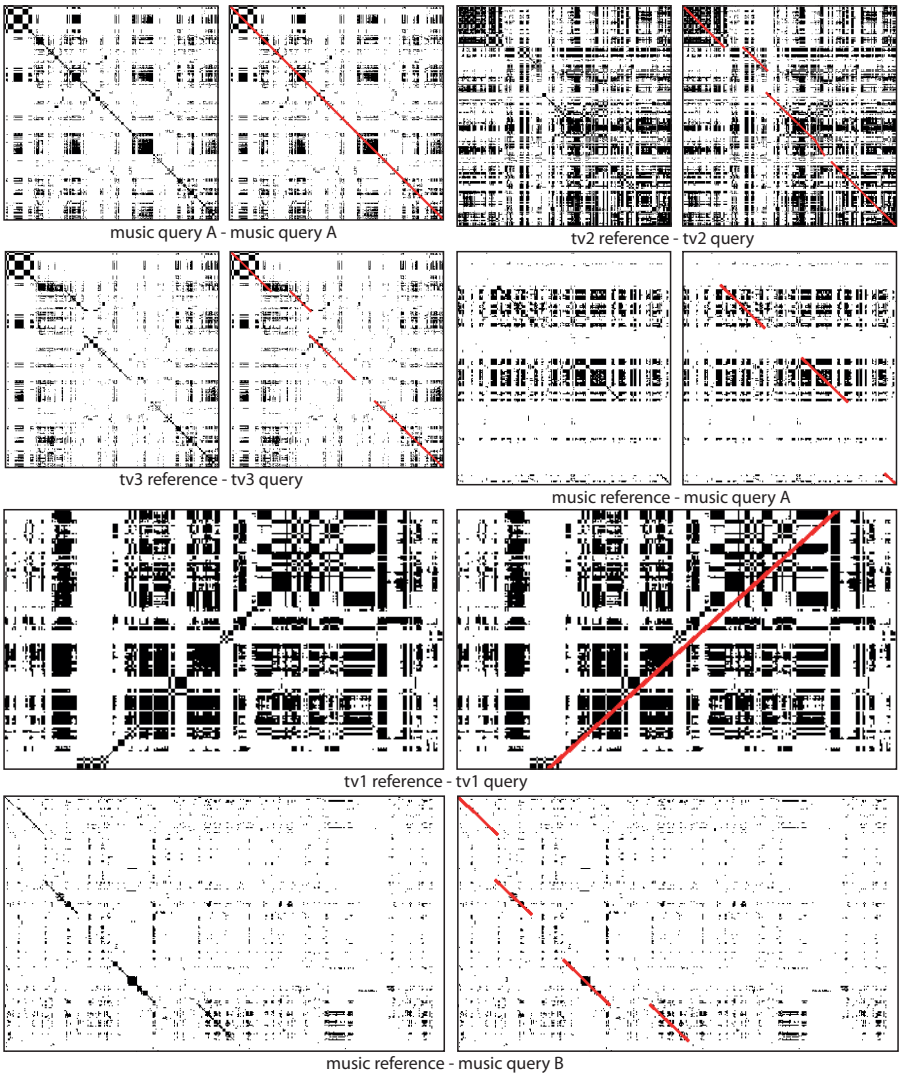


Figure 17: Similarity matrices for evaluation of the matching segments extraction. Due to its proportions, the image for **tv1** is shown rotated.

4.4 Computational speed

The goal of these tests was to evaluate computational performance of the whole process and to determine a proportional representation of its parts. We tested both keyframe extraction methods separately. Tests were made on a common personal computer with Intel Core 2 Duo T7100@1.80Ghz processor, 4GB of RAM and Windows 7 64 operating system, used sequence contains a total of 28358 frames with resolution of 624x352px. Achieved results are shown in Table 2. You can see that the global approach for keyframes extraction is much faster than the local, tracking-based one. Considering that both of them give similar results on most of the supposed input data, we claim the histogram-based approach overall better.

	Global	**Local**
Computation time	164.6s	2136.8s
Framerate	172.4 frames/s	13.3 frames/s

Table 2: Performance of the whole video comparison process.

Notice, that the computational cost only of the video processing steps is measured, so in practical application, also the time for video frames decoding must be taken into account.

The proportional representation of its particular steps is shown in Table 3. It is clear that keyframes extraction, which means browsing all frames of the video sequence, is the most time-consuming part. The proportion changes slightly when using local or global keyframe extraction, due to the massive demands of tracking local features – it simply tremendously beats all other parts. On the other hand, proportion of similarity matrix computation and its segments extraction is practically negligible, which means that future speed optimizations should be done on the first two parts.

	Global	**Local**
Keyframes extraction	75.2%	95.3%
Descriptors computation	23.5%	4.4%
Similarity matrix computation	1.2%	0.3%
Segments extraction	0.1%	¡0.1%

Table 3: Proportional representation of particular steps of the video comparison.

5 Video Matcher Tool

The section describes the particular basic and advanced usage and the information about integration of the methods to Video Terror server platform.

5.1 Command-line version

For demonstration of the introduced algorithm, we developed a simple application with command-line interface. It is implemented in C++ with utilizing of OpenCV 2.4 library for image processing, it can be compiled and used in Windows or Linux operating systems. It requires two input video sequences: One is called reference, which should be the original sequence (without adds, gaps or shifts]; the second one is called test and this the sequence that we want check for differences from the reference one.

5.1.1 Usage and parameters

The command-line interface of the application is simple, it can be run by:

```
vmatch.exe -r ref.avi -t test.avi [-n step] [-t thr.] [-x] [-h]
```

Mandatory parameters:

- `-r reference.avi` – Sets the file name of the reference video sequence.
- `-t test.avi` – Sets the file name of the tested video sequence.

Optional parameters:

- `-n step` – Sets the step between forced keyframes (default 25).
- `-t threshold` – Sets the threshold for distance of two frame descriptors to be considered similar (default 30).
- `-x` – Switches output from simple text to structured XML.
- `-h` or `--help` – Displays simple description of application interface.

5.1.2 Output formats

In the output, information about matching and differing segments from the reference video sequence are provided. The output can be set to one of these two variants:

1. **Simple text output** – it is the prefered way if just several comparisons are done and the results are read by human.
2. **Structured XML output** – common XML parser can be used to read the results, which is useful when other application is used for further processing.

Example of the simple text output:

```
ref/00:00:00-00:00:34 matches test/00:00:00-00:00:34
ref/00:00:35-00:01:04 matches test/00:01:19-00:01:54
ref/00:01:04-00:01:36 differs
ref/00:01:36-00:02:15 matches test/00:02:26-00:03:08
ref/00:02:15-00:02:47 differs
ref/00:02:47-00:03:20 matches test/00:03:08-00:03:42
ref/00:03:20-00:03:43 differs
```

Example of the structured XML output for the same results:

```
<segments>
  <segment start="00:00:00" end="00:00:34">
    <match start="00:00:00" end="00:00:34" />
  </segment>
  <segment start="00:00:35" end="00:01:04">
    <match start="00:01:19" end="00:01:54" />
  </segment>
  <segment start="00:01:04" end="00:01:36" />
  <segment start="00:01:36" end="00:02:15">
    <match start="00:02:26" end="00:03:08" />
  </segment>
  <segment start="00:02:15" end="00:02:47" />
  <segment start="00:02:47" end="00:03:20">
    <match start="00:03:08" end="00:03:42" />
  </segment>
  <segment start="00:03:20" end="00:03:43" />
</segments>
```

If there are some modifications in the test sequence when compared to reference one, it will result into one or several detected differing segments in the output of the application.

5.2 Integration into Video Terror system

The developed and evaluated solution is partially integrated into Video Terror server platform using VTAPI. In VTAPI, every sequence is represented by a class Sequence, which is created by interface of the Dataset class. The integration is based on realized wrapper class for the VTAPI Sequence that adapted its interface to the video sequence iterator in the method's core implementation. The result is that sequences from the Video Terror server database can be compared, the selection of both videos is done by giving the name of the dataset and the name of the concrete sequence. System function is in compliance with the description in former sections – both sequences are preprocessed, similarity matrix is computed and finally, segments extraction is done. The next step of the integration will defined from the results of project Tasks 3.3 and 3.4 together with user-interface design activities.

6 Conclusion

The presented work describes practical video processing application for detection of short- and long-term changes between two video sequences in detail. Two video sequences might be declared as the identical, but small differences might appear. An auto-detection system is needed as human manual comparison in such situation is extremely time consuming. The video-pair dissimilarity analysis is based on detection and validation of the video-frame differences between two visually almost identical video sequences. The solution must be sensitive to re-written, removed or injected video-parts.

The proposed approach is based on modern image and video processing methods that are briefly introduced together with the discussion of their usability for given task. The image processing methods are carefully selected to be robust to several image distortions that might appear in defined task. The overall procedure involves keyframes extraction analysing each video sequence in temporal domain. Then, keyframes from both video sequences are compared each by each and similarity matrix is built. The analysis of similarity matrix then results in candidate video-parts that are validated, classified and reported as the final output of the method.

All parts of the solution have been evaluated and tested on specific dataset particularly created for the purpose of this work to best reflect the task domain. The algorithms are selected and optimized to be effectively integrated into VideoTerror hardware solution. The developed method is the core of the end-user tool. The final application, its usage and parameters, performance and output formats is described in detail. The final test validated the usability and readiness of the solution for practical deployment of the developed solution.

During the development, the application has been presented to the end-users. The outputs of their experiences have been used to improve the practical impact of the application.

References

[1] Bay, H., Tuytelaars, T., and Gool, L. V. Surf: Speeded up robust features. In *In ECCV* (2006), pp. 404–417.

[2] Beran, V. *On-line Data Analysis Based on Visual Codebooks*. PhD thesis, 2011.

[3] Chum, O., Philbin, J., Isard, M., and Zisserman, A. Scalable near identical image and shot detection. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval* (New York, NY, USA, 2007), ACM, pp. 549–556.

[4] J. Geusebroek, R. van den Boomgaard, A. S., and Geerts, H. Color invariance. *PAMI 23*, 12 (2001), 1338–1350.

[5] Jurie, F., and Schmid, C. Scale-invariant shape features for recognition of object categories. *Conference on Computer Vision and Pattern Recognition 2* (2004), 90–96.

[6] Kadir, T., and Brady, M. Scale, saliency and image description. *International Journal of Computer Vision 45*, 2 (2001), 83–105.

[7] Klicnar, L., and Beran, V. Robust motion segmentation for on-line application. In *Proceedings of WSCG'12* (2012), 20-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, University of West Bohemia in Pilsen, pp. 1–6.

[8] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[9] Matas, J., Chum, O., Urban, M., and Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference* (London, UK, September 2002), P. L. Rosin and D. Marshall, Eds., vol. 1, BMVA, pp. 384–393.

[10] Mikolajczyk, K., and Schmid, C. Scale & affine invariant interest point detectors. *International Journal of Computer Vision 60*, 1 (2004), 63–86.

[11] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. A comparison of affine region detectors. *International Journal of Computer Vision 65*, 1-2 (2005), 43–72.

[12] Rosten, E., and Drummond, T. Machine learning for high-speed corner detection. In *In European Conference on Computer Vision* (2006), pp. 430–443.

[13] Sailer, Z. Image retrieval based on color histograms, 2012.

[14] Sivic, J., Schaffalitzky, F., and Zisserman, A. Object level grouping for video shots. *International Journal of Computer Vision 67*, 2 (2006), 189–210.

[15] Sivic, J., and Zisserman, A. Video Google: Efficient visual search of videos. In *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, Eds., vol. 4170 of *LNCS*. Springer, 2006, pp. 127–144.

[16] Tuytelaars, T., and Gool, L. V. Matching widely separated views based on affine invariant regions. *Int. J. Comput. Vision 59*, 1 (2004), 61–85.

[17] Zezula, P., Amato, G., Dohnal, V., and Batko, M. *Similarity Search - The Metric Space Approach*, vol. 32 of *Advances in Database Systems*. Springer, 2006.