Video Sequence Summarization and Synopsis

FIT BUT Technical Report Series

Vítězslav Beran,
Lukáš Klicnar

Technical report no.: FIT-G20102015006-2012–04
Faculty of Information Technology, Brno University of Technology

Last modified: January 12, 2015

Video Sequence Summarization and Synopsis

Vítězslav Beran (`beranv@fit.vutbr.cz`)
Lukáš Klicnar (`iklicnar@fit.vutbr.cz`)

January 12, 2015

Abstract

The technical report presents the evaluation of video processing methods focused on video-sequence summarization based on visual content analysis in the scope of the VideoTerror project. One set of video summarization methods measure the change of visual information (illuminations, gradiets etc.) in whole frames in temporal domain and evaluates the activity score for particular video-parts. The resulting video-sequence then contains only the most interesting parts from input video-sequence. The second set of methods measure the changes of sub-frames (image segments) of video-frames. Extracted interesting active sub-parts of original video-sequence are then rendered into final resulting video-sequence with time-ovelap so the effectivity of summarization process is further improved.

The appropriate computer-vision methods are presented and discussed, then proposed method and experiments are introduced and evaluated on manually annotated 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.

1 Introduction

Video summarization is a process of creating a short video summary from a long video sequence. With increasing amounts of video today, this is a crucial task since a simple analysis may become extremely time consuming and unbearable. Several properties are typically demanded, the summary should be much shorter, and also contain every important parts or even contain only the most important ones. But there may be another demands, that usually depends on a particular application of video summarization. It may involve requirements about duration of every part, continuity, preservation of time order, preference of certain types of objects, or even more complex attributes. Also, the opinion of what is important in video is a personal choice, other people may emphasize different aspects, events, etc. There is simply no single way, how the summary should be created.

In our work, we represent the video sequence as a decomposition into multiple parts – so called key-volumes. Each part is represented by its temporal information (begin and end) and it is also constrained spatially (mask for every frame). The main feature of these key-volumes is that, if they are obtained suitably for this task, they represent significant parts of video, which means they can act as a pre-selector of importance. The summarization algorithm then works with key-volumes and it can basically operate at two levels: The first one is *local*, which means key-volume segmentation and local adjustments. The second level is a *global* arrangement of these units and composition of the resulting summarized video. Of course, these approaches also works well when applied together.

Structure of this report is following: First, the key-volumes extraction step is presented in Section 1. A brief introduction is followed with a description of two different approaches – the first is based on global features and the second one utilizes local features tracking and motion segmentation. Section 2.2.3 then goes through a process of score of importance estimation for these key-volumes. Several metrics are presented, which is important for the next step, the local and global optimization. This may be called the core of summarization and it is described in Section 3. It addresses both a ”classical” summarization approaches and a video synopsis, which can create even more condensed video. Only some preliminary experiments were made, they are described in Section 4.2. Also, several tools were created for demonstration and experimenting purposes, their brief introduction is contained in Section 5.3.

2 Key-Volumes extraction

This section describes the first stage of our summarization algorithm, which results to breaking the video sequence into several spatio-temporal units needed for further processing. These units are called key-volumes and they represent some activity in time and space. The first property means they occupy a continuous time interval in video defined by initial and end frame, the second one means only pixels of some particular object (which is exactly represented by the key-volume) are accounted into computations – this is usually done by adding a pixel-wise mask for every frame. It should be noted that a special case of key-volume is a global one for the whole video sequence, when the mask includes all pixels of the frame and the volume covers time interval from the beginning to the end of the video. This understanding of video as a key-volume enables us to apply the same algorithms for further processing (described in Section 2.2.3 and 3).

In the most general way, these key-volumes can correspond directly with every pixel separately, without their spatio-temporal context (no information about their position in time or space). In this case, compactness of the result must be ensured by more complex further processing, because we are usually interested in viewing e.g. objects rather than separate pixels. Typically, key-volumes are extracted at a higher level, in an optimal case probably corresponding to particular objects. For a summarization task, we consider movement as an approximation of events importance, so following methods for key-volume extraction are based on moving objects detection.

2.1 Image subtraction

The most basic approach for moving objects detection is a simple background subtraction as used in [7]. Difference of value between pixels at the same position in two consecutive frames is defined as an activity measure and this is then thresholded to determine, which parts of frame should be considered dissimilar in time. This dissimilarity is most likely caused by some movement in the video, so this step equals to moving objects detection. Alternatively, there may be several thresholds to segment the frame into multiple areas with different levels of activity (low, middle, high), as authors in [2] did with K-means clustering.

More advanced methods are based on better background models (which is the previous frame in the most simple way), for example pixel-wise median over some longer time interval [8], but the subtraction principle remains the same. Noise can be lowered by Gaussian blur before the subtraction step and by utilizing a morphological dilation and erosion as a post-processing. This results to an activity map, where particular areas usually correspond to moving objects – their tracking can be done by simple overlap-based frame-to-frame correspondence estimation. Of course, this approach can be used with static-camera sequences only, but this proved to be sufficient for typical surveillance camera applications. The main advantage is a very high computational efficiency due to its simplicity.

2.2 Motion segmentation

The second method for obtaining key-volumes is our motion segmentation-based approach [3]. Its main advantage is a great robustness and ability to work with moving-camera data, but it comes with a price of high computational demands. It is based on a sparse feature tracking, the core of the algorithm is a local similarity-based grouping of trajectories, so a coherent groups are obtained, which usually correspond with moving objects. Initial motion segmentation is estimated by a RANSAC-based algorithm, groups are then partitioned and their frame-to-frame correspondence is solved. Finally, the Voronoi tessellation is used to obtain a dense image segmentation (masks of moving objects). A block diagram of the whole system is in Figure 1.

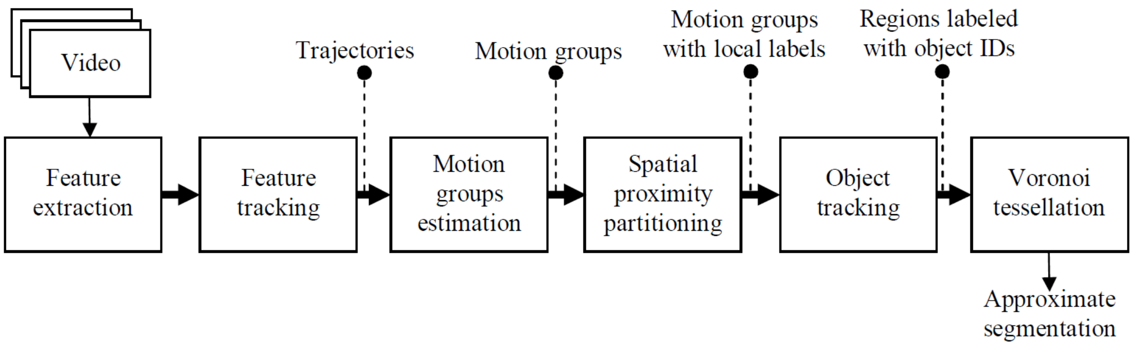


Figure 1: Block diagram of the motion segmentation system.

2.2.1 Feature detection and tracking

One of the most important parts of this approach is a robust tracking method, which provides trajectories as long as possible and ideally without outliers. The length of trajectories has an influence on correct object tracking, while the low occurrence of outliers is essential for good motion segmentation. We found tracking with the KLT tracker [11] based on optical flow very suitable for this task. It doesn't match features from two frames, but it rather searches for a best occurrence in the image directly. In the proposed approach, points provided by the Harris corner detector [4] are used. Features are detected on several different scales, which improves the ability to segment some very fast moving objects, which appear blurred in the original image. In addition, matched features are checked by normalized cross correlation and if it's below a threshold the particular track is terminated. That results to trajectories with no visible outliers. In opposite to [10] [1], there is no need for short-range track repair, we found KLT tracking sufficient.

2.2.2 Motion segmentation

The initial motion segmentation is performed by a RANSAC based algorithm [1] [3]. It progressively extracts groups of features with a similar motion, a one group is extracted in every iteration. Four-tuples of tracks are randomly chosen to estimate a homography matrix that represents the motion between the current and the previous frame. A total reprojection error is then computed, that forms the criterion for suitability of found transformation – the goal is to find the homography corresponding with the lowest reprojection error. This is done iteratively until the maximal number of iterations is reached. Iterating can be stopped also when a desired reprojection error is satisfied. Inlying tracks (with reprojection error below 3.0px) are then removed as a single motion group and the rest of them goes through the same process in a next iteration. Every motion group is then divided to meet the spatial proximity constraint, that the distance between 2 nearest points has to be lower than a threshold, so the group becomes spatially-compact. This may cause an oversegmentation, but this is handled by the next stage, object extraction and tracking.

2.2.3 Objects extraction

From the previous step, every track in a motion group has a local label, which corresponds with a motion segment in a particular frame. For moving objects tracking, the frame-to-frame correspondence of these labels has to be solved. This is done by a one-frame label propagation [3]. First, labels from the previous frame are propagated forwards, into the current frame. For every local label from the current frame, a corresponding label from the previous frame is found – that is the label of the group, which has the highest number of common tracks with the currently processed group. Subsequently, a new, temporary label for each group in the current frame is generated and propagated in the same manner, but in the opposite direction (backwards). Labels are then merged for each pair of groups from the current and the previous frames: The same label (from the forward labeling) is assigned only if both groups correspond together in both directions, otherwise a new one is generated (see Figure 2). New labels aren't assigned immediately to motion groups, but only if the same new label is stably suggested for at least 2-3 frames. This helps to overcome the problem that a motion group can occasionally disappear (tracks are incorrectly assigned to another group with different label), which tends to be stable for max. 1-2 frames.

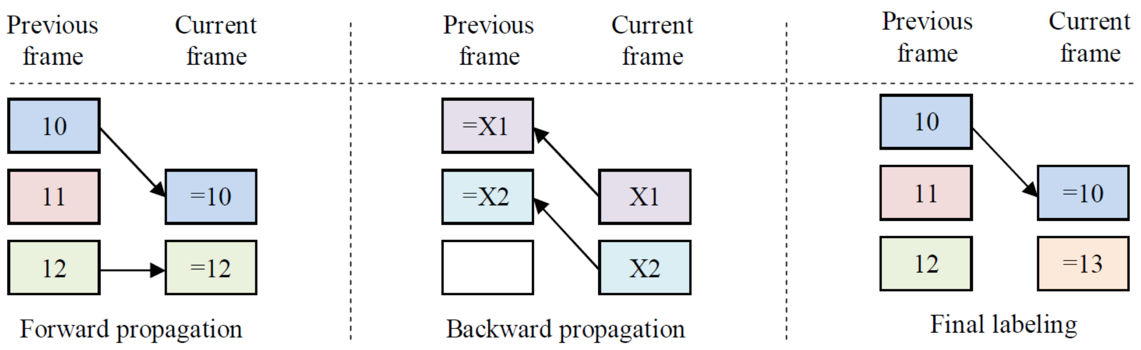


Figure 2: Label propagation.

Because of the sparse approach, masks of moving objects are not directly available from this algorithm's output. To overcome this problem, the Voronoi tessellation is used to obtain an approximate dense segmentation of the whole image [3]. The positions of tracks in the current frame are used as a cell generators, all pixels in each cell are then labeled with the same label as the track that generated the cell. This method creates only a very approximate object borders (Figure 3), but this is not a problem for out summarization task.

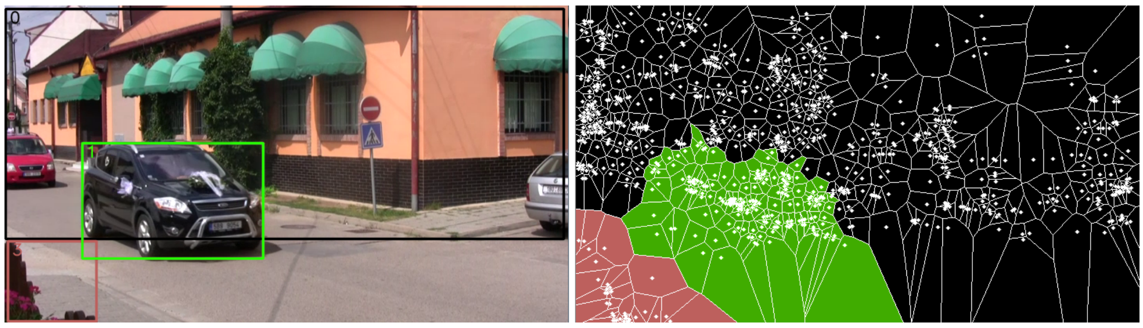


Figure 3: Output of the motion segmentation system.

3 Score of interest for key-volumes

Having video spatial parts (global or key-volumes), we need to determine their significance, as some of them may be more important than others. But this is specific to particular task, application, or even every human personal opinion, so only some approximation is possible. In general, very important property is how objects change in time, so we selected this aspect as our effort for approximation. Many features can be used, we experimented with these following:

- Change of *pixel values* in key-volume – may be caused by object movement, but also by, for example, change of illumination.
- Change of *gradients* in key-volume – approximates object movement, but significant change can be also triggered, for example, by overlapping with another object.

Features are computed for every key-volume (or in the global case, for the whole video sequence). The result is dependent on further processing, it can be done in the following forms:

- *One value* for every key-volume – may be useful e.g. for selection, which volumes should be included in the summarization and which should be omitted.

- *Function* (signal) for every key-volume, which assigns one value to every frame – can be used for segmentation of key-volumes and finding their most important time intervals.

Both of the presented features are computed from every pair of two consecutive frames (although they may contribute to result for the whole key-volume) and the process is also very similar for pixel values or gradients. The difference is only in the first step, in which the input data are prepared. Formally, we have two consecutive frames F_t, F_{t-1} in current time t, key-volume for these frames are spatially constrained by masks M_{F_t}, $M_{F_{t-1}}$, values of a particular pixel at coordinates $\mathbf{p} = (x_p, y_p)$ are $F_t(\mathbf{p})$, resp. $M_{F_t}(\mathbf{p})$, and if pixel at coordinates $\mathbf{p}$ belongs to a key-volume, then $M_{F_t}(\mathbf{p}) = 1$, otherwise 0. The whole process of key-volume importance score computation then consists of these following steps:

1. Intersection of both key-volumes mask for these frames, so only a common pixels contribute to computation in following steps:
$$I_{F_t,F_{t-1}} = \big\{\mathbf{p} | M_{F_t}(p) = 1 \wedge M_{F_{t-1}}(p) = 1\big\}$$
2. Conversion of pixel values of both frames into grayscale (pixel values) $G_t(\mathbf{p}) = \text{grayscale}(F_t(\mathbf{p}))$; or gradient-image (gradients) $H_t(\mathbf{p}) = \text{gradient}(F_t(\mathbf{p}))$ using Laplacian operator. This is why further steps are the same, they only compute with different values V_t:
$$V_t(\mathbf{p}) = \begin{cases} G_t(\mathbf{p}) & \text{if computation with grayscale} \\ H_t(\mathbf{p}) & \text{if computation with gradients} \end{cases}$$
3. Subtraction of both frames:
$$D_t(\mathbf{p}) = |V_t(\mathbf{p}) - V_{t-1}(\mathbf{p})|$$
4. Computation of histogram H_t of values $D_t(\mathbf{p}), \mathbf{p} \in I_{F_t,F_{t-1}}$, divided into n bins (we use $n = 32$) of size s. Value of bin m is $H_t(m)$.
5. Importance score $S(t)$ computation may be done in several ways. Most basicaly, if $I_{F_t,F_{t-1}}$ is empty, or when processing the first frame, $S(t) = 0$. For other cases, value of some histogram bin can be used:
$$S_1(t) = H_t(m)$$
Other functions can be derived from the whole histogram, e.g. maximum-based function can be used:
$$S_2(t) = \underset{m \in <1,n>}{\text{argmax}} \, (H_t(m) \cdot (m-1) \cdot s)$$
We also obtained good results with function:
$$S_3(t) = \sum_{m \in <1,n>} (H_t(m) \cdot (m-1) \cdot s)$$
6. Finally, score values $S(t)$ are normalized:
$$\bar{S}(t) = S(t) / \max_t(S(t))$$

Described computation is for the case, when a function defined for every frame of key-volume is demanded. For a single value for the whole volume, the histogram can be accumulated with all frames, score computation then remains the same. Examples of several different score functions are in Figure 4.

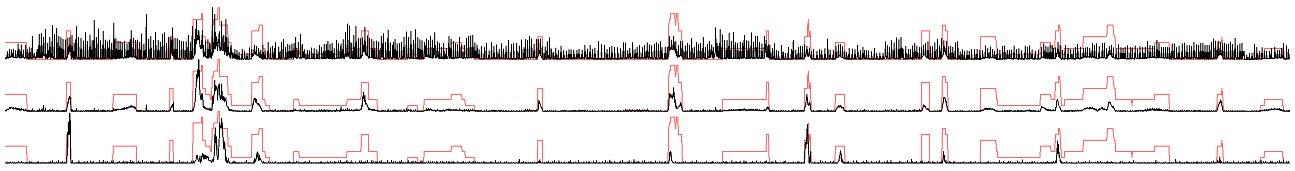


Figure 4: Example of various score functions (black color) and ground truth (red color).

4 Optimization

In the context of video summarization, optimization stage deals with decision, what parts of original data are important and should be present in summarized video, and what parts can be omitted. Its another crucial task is also to determine, how the remaining parts should be arranged both in time and in space. The result is another video sequence (summarized video), which should shorter with more condensed information. This means the whole summarization process is a video to video transformation.

In Section 2.2.3, there was a description of key-volumes extraction – this process forms a pre-selection stage of significant parts, because only possibly-important events can trigger a key-volume creation. But the desired length of summarized video is limited and typically, there is no possibility to include all the key-volumes into a restricted time interval. So a method for best volumes selection and/or segmentation is necessary. There are three possible approaches:

- *Local* – operates on a single key-volume, may result into its segmentation into several key-volumes with different importance.
- *Global* – searches for an optimal arrangement of all key-volumes in time and space. Shortening can be achieved by overlapping them in time, omitting some of them, etc.
- *Combination* of local and global – which means that global approach works on key-volumes with local adjustments.

These aspects can be denoted with a cost function penalizing deviation from theoretically-ideal summarization, which can be treated as an optimization problem.

4.1 Local – selection in temporal domain

This approach works with a single key-volume and its output is basically the same – a set of key-volumes, which is created by segmenting (or just simple shortening) the original one. So it can be also understood as refining. It is useful mainly with the global (whole video) key-volume, then it serves as an importance filter: Resulting set contains only a key-volumes with high importance score.

Our approach segments the key-volume into predefined number n of segments with fixed length of m frames. This is in fact similar to specifying maximal length, which equals $n \cdot m$. The only criterion is the score of importance $S(t)$ (as described in Section 2.2.3)for given key-volume in time t. We developed two methods based on this metric:

The *simple fixed-length interval* approach selects intervals centered around local maxima. In every iteration, strongest local maxima in still uncovered parts of key-volume is found and a new key-volume is created evenly centered around this maxima. This repeats until a desired count of key-volumes is created or the whole original one is covered. This comes with disadvantage of uneven coverage of key-volumes with activity concentrated only in a small part. Local maxima are then likely to be selected from this part, which may leave segments with lower activity uncovered.

The uneven coverage problem is addressed in the second method, we call it *coverage-driven*. It is resolved by introducing a weight function $w(t)$, which is set at 1.0 in the beginning, but gradually lowered in the neighborhood of already selected intervals – so other local maxima located nearby are also suppressed. The algorithm for segments selection then remains the same, only the metric is adjusted with the weight function: $w(t) \cdot S(t)$. Example is in Figure 5.

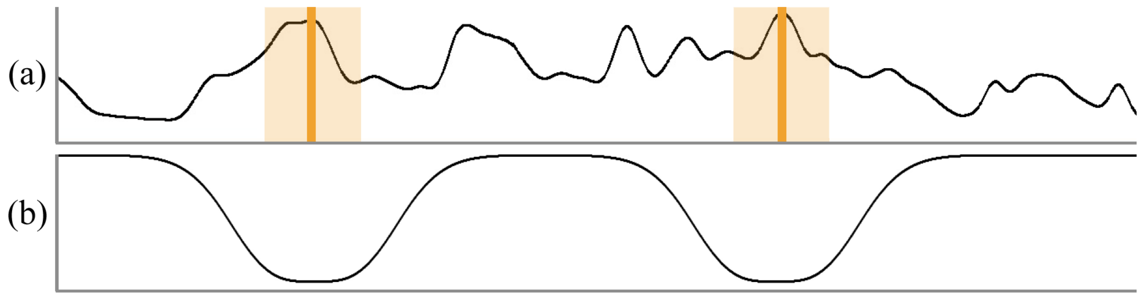


Figure 5: Example of (a) importance score function with already selected intervals, (b) weight function, which is low around the already covered area.

4.2 Global – video synopsis

Having a set of key-volumes, the final task is to arrange them together and render the summarized video. The most trivial (and very often used) solution is a simple concatenation of video parts that correspond to each of involved key-volumes. This is often called a ”classical” summarization, because this practically equals dropping the non-important parts of video.

But there are many ways how to make video content even more condensed: One of them is called video synopsis. It utilizes spatial regions of video with no activity, where another key-volume can be placed by overlapping several of them in time – since every one of them occupy different spatial area in the same time, no collisions emerge. Synopsis algorithm operates with a set of key-volumes and it searches for their optimal rearrangement of key-volumes in time. Their spatial position must remain the same, to be able to blend them properly at the same place on the background without visible artifacts (this also restricts the usage only to a static-camera videos). It should be noted that algorithms with ability of spatial rearrangement also exist [5], but since the key-volumes must be fit into completely new, artificially created background, this does not work automatically in current state of the art.

The core of video synopsis is done by a minimization of the following energy function [7]:

$$E = E_{activity} + E_{collision} + E_{temporal}.$$

As can be seen, the function consists of main three different terms, which covers the main aspects of desired result:

- $E_{activity}$ – activity cost, which makes more important objects more likely to be included. This can be approximated for example by its size, or by its score of importance described in Section 2.2.3.
- $E_{collision}$ – collision cost that is responsible for minimization of spatial collisions between key-volumes with non-empty time overlap. For every pair of key-volumes, it can be approximated by summation of their intersection areas across all frames.
- $E_{temporal}$ – temporal consistency cost tries to preserve the chronological order of original video, which is important, when some of the objects interact together (so they should also be present together in the summarization).

Any general minimization technique can be involved for solving this problem, a simulated annealing method (also in [7] [8] [9]) was used in our experiments. Since only a spatially-constrained area of every key-volume is used for summarized video composition, they have to be rendered over some background to fill the missing pixels. We used the original video, median-filtered over a relatively long period (5 minutes), which successfully erases moving objects but does not suppress the long term dynamics of the whole scene (e.g. the environment light conditions changes, day/night, etc.). Also, for a blending of key-volumes on this background without visible seams, a Poisson image editing [6] technique is well applicable.

5 Results

This section describes the evaluation process of proposed algorithm. First, our dataset it introduced with simple characteristics of each video sequence, then a description of how annotation was made follows. A simple evaluation was done, brief overview of achieved results is also presented.

5.1 Dataset

For experiments during algorithm development and demonstration, we created a small sample dataset. It consists of five video sequences obtained by a static cameras, every one of them represents a different domain with certain characteristics. The dataset contains these video sequences (example in Figure 6):

- *Cars 1* – typical webcam footage, activity is very sparse and it appears in shorts intervals, for example when a car passes by. It has length of 30 minutes.
- *Cars 2* – shorter version of cars 1 sequence, first 10 minutes.
- *Street* – shot of some crime scene, activity is made mainly by people moving around, many of them is present at the same time. Length of 10 minutes.
- *Conference* – 10 minutes of record of some conference, a little activity is present constantly with sudden other events (for example, some person lefts or enters the room).
- *Nature* – contains record of a group of very slow moving animals (elephants), important events are very difficult to recognize. Length of 10 minutes.

To be able to do some automatic evaluations and see, how algorithm works, we also had to obtain ground truth data for every of these sequences, which means we created a human-made annotation with estimated importance of every frame. For this, we implemented a simple annotation tool. User can step frame-to-frame through the whole video sequence and for every frame, a selection of perceived importance is made. This is done by pushing numbers 1-9 on keyboard. For better precision, it is also possible to step back in video and correct the score. It could be also computed as an average from several annotations made by different people, which should lead to more general description. In our case, the importance of every frame was estimated primarily according to subjective significance of movement in the scene (speed, relative size, presence of multiple movements, etc.).

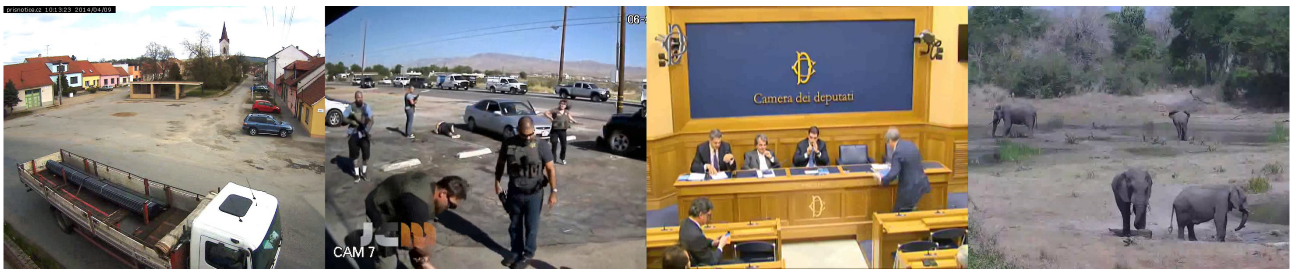


Figure 6: Examples of video sequences in dataset – cars, street, conference, nature.

5.2 Evaluation protocol

Due to the subjective manner of certain video parts importance, any evaluation of summarization quality is very complicated. We conducted only some basic experiments based on annotations described in Section 5. The goal was to determine, how cumulative importance score of frames included in summarization compares to a maximum potential, if same number of frames were selected based on annotation score known apriori. If we denote a set of frames F (which is a subset of all frames of a video sequence), we can create two different sets:

- F_{summ} – contains all frames (their count is m) included in summarization.
- F_{max} – is created by taking the top m frames with highest annotation score. Frames are taken separately without any time continuity and since only their count and score is required in further computations, possible existence of multiple frames with the same score is not an issue, they can be selected e.g. randomly.

Now, a cumulative score $S(F)$ over a set of frames F is defined:

$$S(F) = \sum_{f \in F} score(f),$$

where $score(f)$ is a function that assigns every frame its score of importance. Two cumulative score functions are defined, since the score may come from algorithmic computation (Section 2.2.3, function $S_s(F)$) or from annotation ($S_a(F)$). But this should be also easily known from context and in this case of evaluation, annotation will be used exclusively.

Finally, an *importance score ratio* ISR is defined:

$$ISR = \frac{S_a(F_{summ})}{S_a(F_{max})},$$

which is the desired metric presented in the beginning of this chapter. This was used to evaluate the performance of the summarization method. Since several possible variations were introduced in the previous sections, the results were obtained for the following combinations of methods and features for score of importance estimation (all described in Sections 2.2.3 and 3):

- *FLv* – simple fixed-length interval method, importance score S_3 computed from pixel values.
- *FLg* – simple fixed-length interval method, importance score S_3 computed from gradients.
- *CDv* – coverage-driven method, importance score S_3 computed from pixel values.
- *CDg* – coverage-driven method, importance score S_3 computed from gradients.

In addition, several different summarization target lengths were evaluated (1 %, 2 %, 5 %, 10 %, 15 %, 20 %, 30 %, 50 %) for every video of the dataset. The length is defined as the appropriate number of segment with 2 seconds duration.

5.3 Results

Results obtained by the described evaluation protocol are shown in the following tables. The overall results – cumulated over all video sequences from the dataset (Table 1) – definitely proved better performance of grayscale values feature than the gradient-based one. For a summarizations of short length (which is probably close to a real use case), both fixed-length and coverage-driven methods give similar results, but the difference becomes substantial with increasing target length. However, this is expected behavior, since from the coverage-driven method definition, the importance of selected frames is partially downtraded for a more even video coverage. That implies also frames from less important parts are included, which results to a lower cumulative score of importance. The decreasing ISR values for longer summaries are explainable for a similar reason: Because the frames are not selected separately, but with their surrounding frame interval, there are usually some low scored frames also included. This is in opposite to the ISR metric, which selects every frame separately with no temporal continuity consideration.

	1 %	2 %	5 %	10 %	15 %	20 %	30 %	50 %
FLv	0.8	0.79	0.75	0.72	0.73	0.71	0.67	0.62
FLg	0.63	0.68	0.69	0.65	0.6	0.58	0.57	0.51
CDv	0.80	0.75	0.71	0.63	0.53	0.43	0.37	0.37
CDg	0.63	0.68	0.60	0.56	0.51	0.44	0.39	0.39

Table 1: Overall results – ISR values for different summarization lengths and methods.

If we look at the results for each video sequence separately (Table 2), some of them are significantly worse than the others. For example, the *nature* sequence has very low ISR values for all method and feature combinations. This can be explained by a character of activity distribution in the sequence, which is very low, constant and no significant events can be found even when observed by people. So also the annotation is not very reliable. But if the activity in video is defined more clearly, the methods perform quite well.

	1 %	2 %	5 %	10 %	15 %	20 %	30 %	50 %
FLv								
cars 1	0.92	0.87	0.90	0.84	0.83	0.81	0.80	0.80
cars 2	0.81	0.89	0.80	0.78	0.73	0.75	0.72	0.64
street	0.83	0.63	0.73	0.68	0.71	0.69	0.69	0.66
conference	1.00	1.00	0.82	0.76	0.81	0.76	0.64	0.49
nature	0.44	0.44	0.34	0.38	0.46	0.48	0.51	0.55
FLg								
cars 1	0.92	0.90	0.92	0.80	0.80	0.81	0.73	0.68
cars 2	0.85	0.92	0.82	0.67	0.68	0.68	0.67	0.64
street	0.43	0.64	0.72	0.75	0.73	0.72	0.75	0.68
conference	1.00	1.00	0.90	0.77	0.58	0.48	0.40	0.32
nature	0.00	0.00	0.17	0.32	0.31	0.35	0.39	0.37
CDv								
cars 1	0.92	0.87	0.83	0.77	0.70	0.62	0.58	0.58
cars 2	0.81	0.89	0.79	0.73	0.65	0.61	0.61	0.61
street	0.83	0.63	0.78	0.72	0.61	0.48	0.41	0.41
conference	1.00	1.00	0.70	0.54	0.40	0.31	0.26	0.26
nature	0.44	0.38	0.49	0.43	0.35	0.28	0.24	0.24
CDg								
cars 1	0.92	0.90	0.87	0.83	0.77	0.68	0.64	0.64
cars 2	0.85	0.92	0.78	0.73	0.71	0.71	0.71	0.71
street	0.43	0.64	0.72	0.74	0.61	0.51	0.42	0.42
conference	1.00	1.00	0.54	0.37	0.27	0.21	0.17	0.17
nature	0.00	0.00	0.17	0.23	0.34	0.31	0.29	0.29

Table 2: ISR values for selected methods and summarization lengths, each video sequence separately.

6 Video Summarization Tool

This section describes tools we created for the summarization algorithm demonstration. Two applications were created: The simple one is a command-line version of summarizer (Section 6), which is more suitable e.g. for batch processing, but it doesn't work much interactively. For interactive visualization, we provide an application with graphical user interface (Section 6.1). Both tools are implemented in C++ with utilizing of OpenCV 2.4 library for image processing, the second one also uses Qt library for the graphical user interface. They can be compiled and used in Windows or Linux operating systems.

6.1 Command-line version

As described in previous chapters, the summarization is created in two stages, which is also reflected in the design of our experimental tools. First, the key-volumes extraction and computation of their scores of interest must be done, which is realized by a *FeatureExtractor* application. Its usage and parameters are following:

```
FeatureExtractor.exe -t tubes_dir video.avi | -v video.avi
```

- `tubes_dir` – Path to the directory with extracted tubes (key-volumes).
- `video.avi` – Input video sequence.

It can work either with tubes or the complete video sequences, computed data are stored in a CSV (comma-separated value) file in the same directory as the input data (tubes or video). With this tool, all the time-consuming video preprocessing is done, so a summarization can be computed quickly with a *Summarizer* application. Besides the video sequence, it works with the CSV output of the previous tool and its result is finally the summarized video. It can be used in a following way:

```
Summarizer.exe feature.csv feature_index segment_length max_segments_count
in_video.avi out_video.avi
```

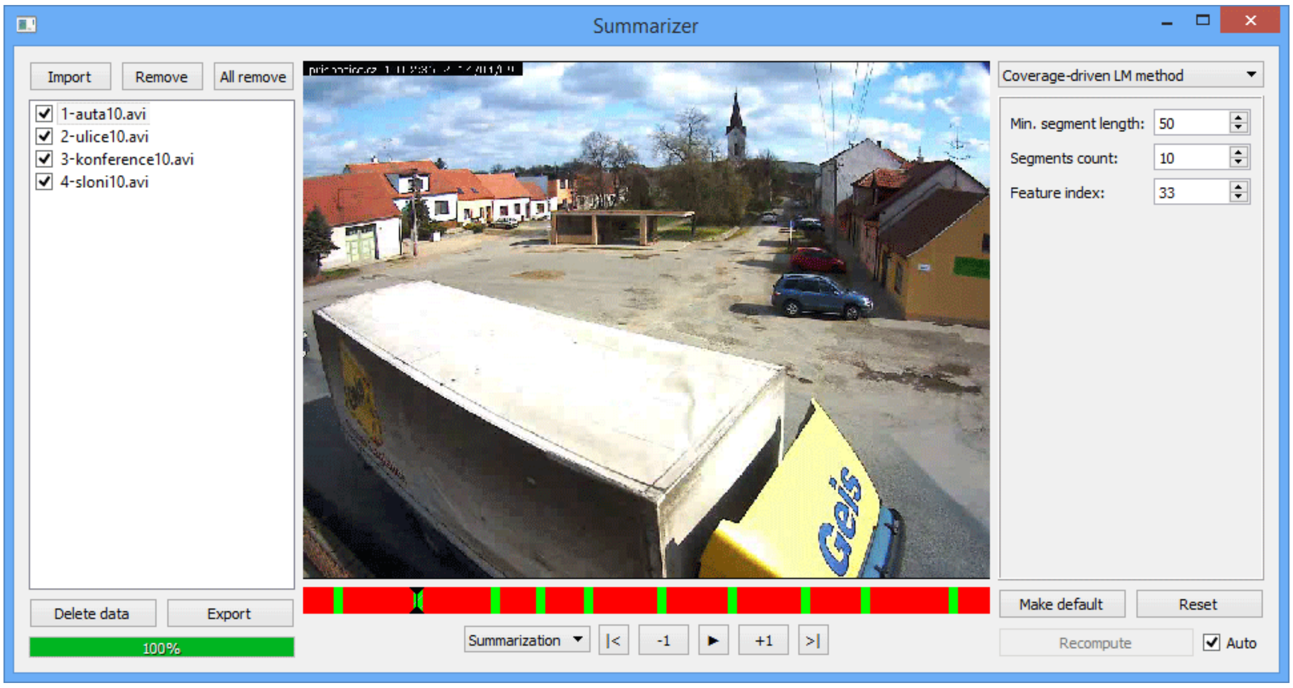
- `features.csv` – Extracted features obtained from FeatureExtractor.
- `feature_index` – Select one feature of features.csv to work with.
- `segment_length` – Length of video segments that the summarization consists of.
- `max_segments_count` – Maximum number of segments to use.
- `in_video.avi` – Input video filename.
- `out_video.avi` – Output summarized video filename.

Because the CSV file contains multiple features, it is necessary to specify, which one will be used as a score of interest. The optimization is constrained by a fixed count of segments with predefined length, so these parameters also have to be specified.

6.2 Interactive Analysis Tool

The second tool is a standalone desktop application, which basically combines functionality of both previous command-line tools together and most importantly, it offers interactive graphical visualization. That means the user can adjust parameters of the algorithm and immediately see the change in resulting summarization. But it requires interaction, so it cannot be used for batch processing.

Main window of this application (Figure 7) consists of three different parts, which are basically arranged in columns. The first one is a video selector, which allows to import one or multiple video sequences to work with. Preprocessing has to be run manually with a button in the bottom part, the computation is indicated by a progress bar and signalized with a checkbox, when it is done. The middle part is used for summarization visualization and video playback. On the timeline, green parts will be included in the summarized video, red parts will be discarded. Finally, the most right column allows to change parameters of the summarization algorithm.



1-auta10.avi ✓
2-ulice10.avi ✓
3-konference10.avi ✓
4-sloni10.avi ✓

100%

Coverage-driven LM method

Min. segment length: 50
Segments count: 10
Feature index: 33

Auto ✓

Figure 7: Main window of interactive analysis tool for summarization.

7 Conclusion

The goal of this technical report was to describe our progress in the field of video summarization. First, the concept of key-volumes was introduced with a two methods of extraction and several metrics for their significance estimation. The proposed summarization approach is based on decomposition of video sequence into these key-volumes, a global and local techniques were introduced. They are able to create summarized videos with different properties, from a simple non-important frame intervals omitting, to spatially compact video synopsis by even bringing together objects from different times. Algorithms for the practical solution were selected with prior to expected data domain and VideoTerror platform, so a high computational performance and simplicity was prefered, which naturally lead us to utilization of the global approaches.

Only some preliminary experiments were conducted on the current solution. It proved to be capable of creating a short summary, but since there is no objective metric of evaluation and no simple way to determine, if the summarization is good or bad, the quality of important parts selection step is a big question. A proper way would be to prepare a set of test scenarios and measurable objectives, which would be evaluated on many respondents observing different summarized videos. But due to expensiveness of such testing, we consider our experiments currently sufficient.

The proposed solution was implemented in a form of demonstration applications. The first one has a command-line interface, it is suitable for batch processing. For more interactive work, also an application with graphical user interface was developed. Both of them can be compiled and used in Windows or Linux operating systems.

References

[1] BASHARAT, A., ZHAI, Y., SHAH, M. Content-Based Video Matching Using Spatiotemporal Volumes. In *Computer Vision, Image Understanding*, vol. 110, 2008, pp. 360–377.

[2] KANG, H., CHEN, X., MATSUSHITA, Y. Space-Time Video Montage. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision, Pattern Recognition*, vol. 2, 2006, pp. 1331–1338.

[3] KLICNAR, L., BERAN, V. Robust Motion Segmentation for On-line Application. In *Proceedings of WSCG'12*, ZCU v Plzni, 2012, pp. 205–212.

[4] MIKOLAJCZYK, K., SCHMID, C. Scale & affine invariant interest point detectors. *International Journal of Computer Vision 60*, 2004, 63–86.

[5] NIE, Y., XIAO, C., SUN, H. Compact Video Synopsis via Global Spatiotemporal Optimization. In *IEEE Transactions on Visualization, Computer Graphics*, vol. 19, no. 10, 2012, pp. 1664–1676.

[6] PEREZ, P., GANGNET, M., BLAKE, A. Poisson Image Editing. In *Proceedings of ACM SIGGRAPH*, 2003, pp. 313–318.

[7] RAV-ACHA, A., PRITCH, Y., PELEG, S. Making a Long Video Short: Dynamic Video Synopsis. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision, Pattern Recognition*, pp. 435–441.

[8] RAV-ACHA, A., PRITCH, Y., PELEG, S. Nonchronological Video Synopsis, Indexing. In *Journal IEEE Transactions on Pattern Analysis, Machine Intelligence archive*, vol. 30, no. 11, 2008, pp. 1971–1984.

[9] RODRIGUEZ, M. CRAM: Compact Representation of Action in Movies. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision, Pattern Recognition*, 2010, pp. 3328–3335.

[10] SIVIC, J., SCHAFFALITZKY, F., ZISSERMAN, A. Object Level Grouping for Video Shots. In *International Journal of Computer Vision*, vol. 67, 2006, pp. 189–210.

[11] TOMASI, C., KANADE, T. Detection and Tracking of Point Features. In *Technical Report CMU-CS-91-132*, Carnegie Mellon University, 1991.