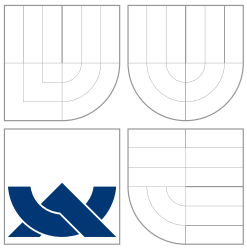
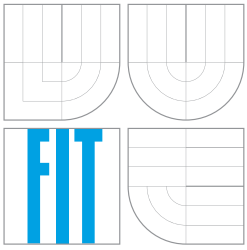


BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

On-line Data Analysis Based on Visual Codebooks

On-line Analýza Dat s Využitím Vizuálních Slovníků

DOCTORAL THESIS
DISERTAČNÍ PRÁCE

AUTHOR — Ing. Vítězslav Beran
AUTOR PRÁCE

SUPERVISOR — Doc. Dr. Ing. Pavel Zemčík
VEDOUCÍ PRÁCE

BRNO 2010

Abstract

This work introduces the new adaptable method for on-line video searching in real-time based on visual codebook. The new method addresses the high computational efficiency and retrieval performance when used on on-line data.

The method originates in procedures utilized by static visual codebook techniques. These standard procedures are modified to be able to adapt to changing data. The procedures, that improve the new method adaptability, are dynamic inverse document frequency, adaptable visual codebook and flowing inverted index. The developed adaptable method was evaluated and the presented results show how the adaptable method outperforms the static approaches when evaluating on the video searching tasks.

The new adaptable method is based on introduced flowing window concept that defines the ways of selection of data, both for system adaptation and for processing. Together with the concept, the mathematical background is defined to find the best configuration when applying the concept to some new method.

The practical application of the adaptable method is particularly in the video processing systems where significant changes of the data domain, unknown in advance, is expected. The method is applicable in embedded systems monitoring and analyzing the broadcasted TV on-line signals in real-time.

Keywords

on-line data analysis, real-time video processing, adaptive visual codebook, image local features

Bibliographic citation

Vítězslav Beran: *On-line Data Analysis Based on Visual Codebooks*, Doctoral thesis, Brno, Brno University of Technology, Faculty of Information Technology, 2010

Contents

Chapter 1

Introduction

This work focuses on processing of video sequences, extraction of features describing the content of the video sequences, and search in the video sequences based on the features describing the content using visual codebooks.

The visual codebook can be seen as the set of image patterns, often accompanied also by the pattern weight representing its entropy. The image then can be described as the list of such image patterns appearing in the image. This presentation is convenient for many computer vision tasks - image retrieval, image classification, image matching, etc.

The image processing pipeline based on visual codebooks are mainly composed from several components (Fig. 1.1). The local visual features are extracted from the images and described using high-dimensional descriptors. Given a pre-trained visual codebook, the high-dimensional descriptors are translated to several most appropriate *visual words*. The images are then represented as the distribution of these visual words. Such representation is called *bag-of-words* or *image signature*.

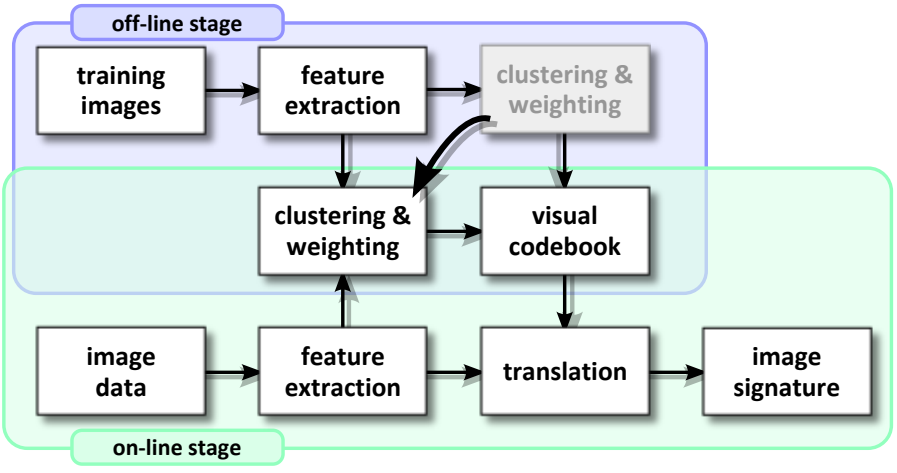


Figure 1.1: Components of image processing pipeline based on visual codebooks.

The visual systems dealing with on-line data, for example monitoring the broadcasted TV signals, are specific in the matter of accessing the data from the time point of view. Such systems process endless data stream and in a particular present

time, the character of upcoming data is not known. Taking into account the data variability, utilization of static visual codebook might be deficient.

The goal of this work is - **design and evaluation of the adaptable method for on-line video searching in real-time based on visual codebooks**. The general state-of-the-art visual codebook techniques are discussed, analyzed and optimized for real-time processing. The modifications of these methods are introduced addressing the high performance on on-line data. In Fig. 1.1, the changes in the static approach are shown by moving the *clustering & weighting* block from the off-line into the on-line stage.

The partial goals of the work are:

- **acquiring datasets** - find appropriate data collections (images, videos, etc.), obtain the access to them and store and preprocess them for evaluation putposes;
- **acquiring the partial methods for image description and clustering** - select state-of-the art image feature extraction methods and clustering approaches and prepare them for experiments;
- **analysis of existing methods** - study the methods focusing on their potentials of utilizing them in real-time approach;
- **evaluation framework for the state-of-the art methods** - design and realization of the framework, run experiments with different image feature extractors and descriptors, experiment with various configurations of clustering methods and weighting schemes;
- **research and realization of novel adaptable method** - modification of the static approach based on visual codebooks addressing the adaptability of the method;
- **experimental evaluation of the novel method** - run experiments with various video data transformations to reveal the ability of the novel method and to compare it with the static approach;
- **definitions of metrics** - introduction of metrics for evaluation of image retrieval tasks and metric evaluating the video search task.

This paper is focused on the description of the design, realization and experimental evaluation of the novel method for the on-line data analysis based on visual codebooks.

Chapter 2

State-of-the-Art Approaches

Real-time on-line video data analysis introduces specific demands on the commonly used techniques. For the real-time approaches, the performance of the method could decrease at the expense of the lower computational cost.

2.1 Local Image Features

Detected local image features are expected to be invariant to geometric and illumination changes. Different detectors emphasize different aspects of invariance, resulting in keypoints of varying properties and sampled sizes.

Lowe proposed the method called SIFT (Scale-Invariant Feature Transform) [8] that is sensitive to blob-like structure and is invariant to scale and orientation. The method also describes the local regions using histogram of gradients and is often used only as a descriptor for different image local feature detectors. The blob-like structures can be also detected as a local extremum of a Hessian matrix. Bay et al. [2] effectively approximated the approach based on Hessian matrix by block filters. The SURF (Speeded Up Robust Features) detector is based on effective platform computation of Haar-wavelets on integral images. The authors also introduced new descriptors utilizing the same platform.

The approach known as FAST (Features from Accelerated Segment Test) corners [16], designed by Rosten and Drummond, employs machine learning to construct a corner detector that outperforms all know approaches in the speed point of view. The FAST itself is neither invariant to scale nor to shear. In this work, detected corners are described using a gradient distribution of the region around the detected point. Gradient distribution can be described by HOG (Histogram of Gradients) [4]) or GLOH (Gradient Location and Orientation Histogram) [10]. The FAST detector combined with the GLOH or HOG descriptor is not rotational invariant (labeled as FHOG). This work utilizes the modified rotational invariant GLOH (labeled as FOGH).

Several other scale-invariant interest point detectors have been proposed. Examples are the salient region detector proposed by Kadir and Brady [7], which maximizes 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. [6] or detector by Tuytelaars and Van Gool [18] have the interesting performance. They

seem less amenable to acceleration though. The last detector that is not included in this work but is worth to mention for its promising performance is the region-based detector developed by Matas et al. called Maximally Stable Extremal Regions [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.

2.2 Visual Codebooks

A comparison between two images is fundamental operation in many computer vision applications. 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, e.g. image retrieval, such time complexity is 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. The approach is based on the Vector Space Model [20] 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 is conveyed by the used words. Each dimension corresponds to a separate word (Eq. 2.1). If a word occurs in the document, its value in the vector is non-zero. The similarity of two documents $\mathbf{q}$ and $\mathbf{d}$ is computed by *normalized scalar product*, also known as *cosine distance* (Eq. 2.2). Without the normalization, longer documents are more likely to be found relevant simply because they have more visual words which increases the likelihood of a match.

$$\mathbf{d} = (w_1, w_2, ..., w_K) \tag{2.1}$$

$$\begin{aligned} sim^{cos}(\mathbf{q}, \mathbf{d}) &= \frac{\mathbf{q}.\mathbf{d}}{|\mathbf{q}|.|\mathbf{d}|} \\ &= \frac{\sum_{i=1}^{K} q_i d_i}{\sqrt{\sum_{i=1}^{K} (q_i)^2 \sum_{i=1}^{K} (d_i)^2}} \end{aligned} \tag{2.2}$$

where $||$ is the vector magnitude and K is the size of the visual codebook.

In computer vision, the words, resp. visual words might be obtained from the feature vectors by a quantization process. The objective is to apply vector quantization on the descriptors and subdivide them into clusters whose labels then represent the visual words usually utilizing *k-mean* algorithm. Several solutions were introduced to avoid the limitations of slow sequential methods for projection of the descriptor into the codebook.

The sequential exact computation can be replaced by an approximate nearest neighbor method. One of convenient structures organizing samples in k-dimensional space is a *kd-tree* structure. The *approximate nearest neighbor* avoids prolonged search by limiting the number of leaf nodes we are willing to examine, and setting for the best neighbor found up to that point. Philbin et al. designed *approximate k-means* [13] based on a random forest of extremely randomized trees. Other approach

dealing with large codebooks uses *hierarchical k-means* was introduced by Nister and Stewenius [12].

Let $\mathcal{C}$ be the visual codebook defined as the vector of visual words $\mathbf{v}_k$ represented by the points in the descriptor space of dimensionality *dim* (Eq.2.3).

$$\mathcal{C} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K); \mathbf{v} = (x_1, x_2, \ldots, x_{dim}) \in \mathbb{R}^{dim} \tag{2.3}$$

The visual words are then weighted using *term frequency - inverse document frequency* weights (Eq. 2.4) usually used in text retrieval. The *term frequency* reflects the entropy of a word with respect to each document unlike *inverse document frequency* down-weights words that appear often in the database and vice versa.

$$\begin{aligned} w_{ij} &= tf_{ij} \times idf_i \\ &= \frac{|\mathbf{d}_j(\mathbf{v}_i)|}{|\mathbf{d}_j|} \times \log \frac{|\mathcal{D}|}{|\mathcal{D}(\mathbf{v}_i)|} \end{aligned} \tag{2.4}$$

where $|\mathbf{d}_j(\mathbf{v}_i)|$ is the number of occurrences of the visual word $\mathbf{v}_i$ in document $\mathbf{d}_j$, $|\mathbf{d}_j|$ is the total number of visual words in the document j, $|\mathcal{D}|$ is the total number of documents in dataset and $|\mathcal{D}(\mathbf{v}_i)|$ is the number of documents which contain visual word $\mathbf{v}_i$.

The quantization provides a very coarse approximation to the actual distance between two features - zero if assigned to the same visual word and infinite otherwise. The *soft-assignment* techniques [5, 14] assign a single descriptor to several visual words nearby in the descriptor space. One of the soft-assignment weighting functions is the exponential function (Eq. 2.5).

$$W_l^{exp}(\mathbf{z}) = \exp(-\frac{z_l^2}{2\sigma^2}) \tag{2.5}$$

where $\mathbf{z}$ is a vector of distances of the sample to the k nearest visual words, $l \in \{1, \ldots k\}$ and σ should be chosen so that a substantial weight is only assigned to a small number of visual words.

When using *soft-assignment*, the tf is then computed simply using normalized weight value for each visual word. Let W_i is the set of weights computed from all samples in the document for the visual word i. The tf is then computed as the sum of the weights in W (Eq. 2.6).

$$tf_i^{soft} = \sum W_i \tag{2.6}$$

Chapter 3

Adaptable Method for On-line Data Analysis

The on-line processing of data stream is specific in the matter of accessing the data from the time point of view. In a particular *present* time, the processing system in service is aware only of data in the past and has no clue about the upcoming data.

Usually, the higher level computer vision methods work with models that are pre-trained on some training data of the application domain. In the off-line video processing, the parameters of the models or processing methods can be evaluated from the processed video data. Working with on-line data, when one does not see the upcoming data, the solution must be able to use only already seen data and be adaptable to the new upcoming ones.

This work introduces a novel adaptable method for on-line video searching in real-time based on visual codebooks. The static approach is modified to improve the ability for better reflection of the changes in character of processing data. The pipeline of the system is slightly modified so its able to adapt the visual codebook to actual data characteristics (Fig. 1.1).

3.1 Flowing Window Concept

This section introduces a *flowing window concept* (Fiq. 3.1) that is designed to solve the on-line approach limitations by processing only a part of a data stream. The concept defines two types of data blocks, one used for preprocessing and one for processing. The *training stage* works with the reference block, the query block or both (depending on the application and data accessability) and does all the training computation. The *processing stage* then use the trained models and other structures and process the data within the query block.

Let $\mathcal{D}$ be the set of all already seen frames in the video stream and $\mathcal{D}_\mathcal{B}$ be the subset of frames of $\mathcal{D}$. Then $\mathcal{D}_\mathcal{B}$ is defined as follows:

$$\mathcal{D} = \{\mathbf{d}_j\}; j \in \{0, \ldots, t\} \tag{3.1}$$

$$\mathcal{D}_\mathcal{B} = \{\mathbf{d}_j | \mathbf{d}_j \in \mathcal{D} \wedge j \in \{t - N, \ldots, t\}\} \tag{3.2}$$

Three types of realization of the flowing window concept are suggested - *simple flow*, *large-step flow* and *block-based flow* . The schemas differ in the realization

of training and processing stages as various image processing or computer vision techniques differs in adaptation ability and data accessability demands. The variability of the image processing or computer vision techniques makes impossible to generally decide which schema is best. To be able to evaluate the convenience of the schema for particular techniques, the computational cost for each particular schema is defined.

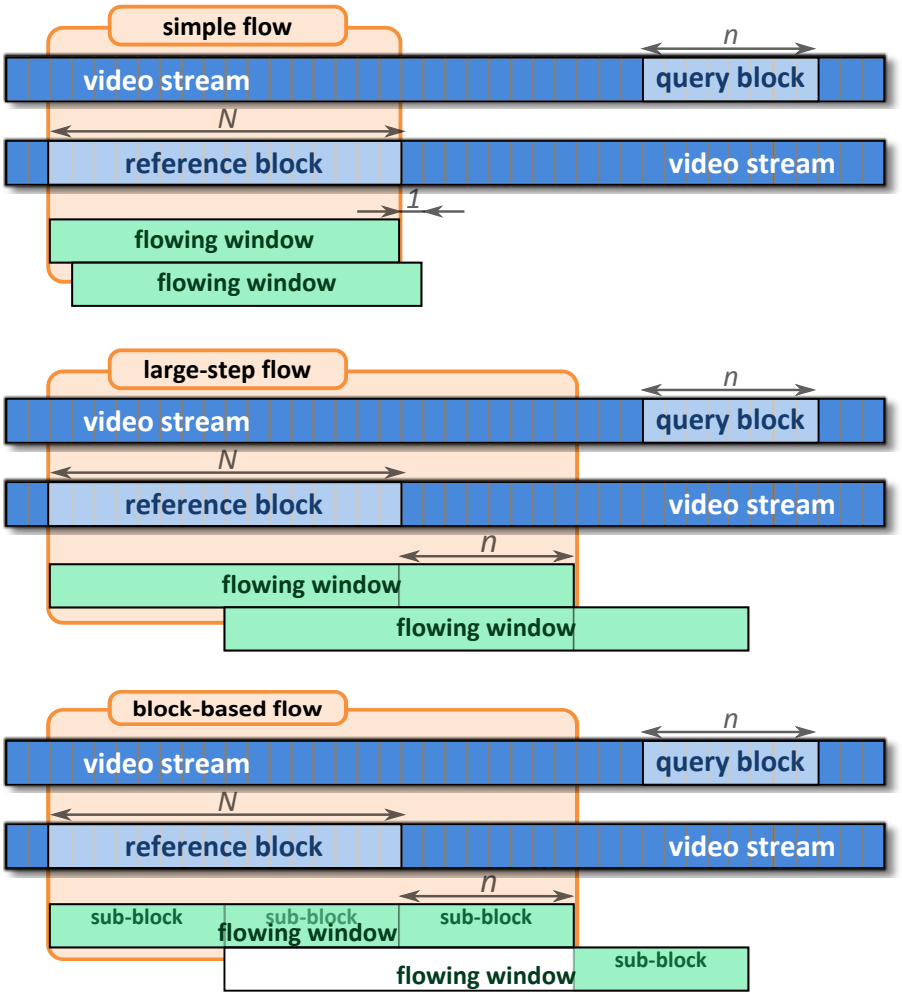


Figure 3.1: Flowing window concept realization schemas.

As the concept defines two stages - training and processing stage - each stage has different cost function for each of all three schemas. The functions f_T (Eq. 3.3), resp. f_P (Eq. 3.4) represent the time complexity of the training stage, resp. processing stage. Two main parameters of the functions are N_T - number of the training samples and N_P - number of the query samples. The parameters P^* and T^* represents the general costs of elementary operations. As they are unit-less values, they can be omitted in some mathematical evaluations. The training c_T and processing c_P costs

are normalized to represent the cost of processing one frame.

$$f_T(N_T, N_P) \tag{3.3}$$
$$f_P(N_T, N_P) \tag{3.4}$$

Although the *flowing window concept* is a general concept, this work assumes the forward flowing of the window on the video signal.

Simple Flow. The *simple flow* approach executes the training stage for each query sample separately. For each new query sample, the reference block is actualized - shifted by one sample - and all data in the block are used for training. The cost of processing of N_Q query samples is N_Q times cost of training stage (Eq. 3.5) and N_Q times cost of processing stage (Eq. 3.6).

$$c_T^{simple} = f_T(N_R, 1).T \tag{3.5}$$
$$c_P^{simple} = f_P(N_R, 1).P \tag{3.6}$$

Large-step flow. The schema improves the processing costs by enlarging the reference block by the N_Q samples. Enlarging the reference block of data enables to execute the training stage just ones for N_Q query samples. The training stage is computed on $(N_Q + N_R)$ data samples (Eq. 3.7) and the trained structures are used to process all N_Q query samples (Eq. 3.8).

$$c_T^{large} = \frac{1}{N_Q} f_T(N_Q + N_R, N_Q).T \tag{3.7}$$
$$c_P^{large} = f_P(N_Q + N_R, N_Q).P \tag{3.8}$$

Block-based flow. The block-based schema also uses the idea to compute the training stage in the way that it could be used to process all N_Q query samples without repetitive re-computation of training stage. The difference is that the reference block is divided into k sub-blocks of the length N_Q: $k = \frac{N_R}{N_Q} + 1$. The training stage is then computed only for each sub-block separately. Two possible cases exist, how to use the trained sub-blocks.

A - it is possible and beneficial to concatenate the trained sub-blocks and then do the processing on one large block:

$$c_T^{block.A} = \frac{1}{N_Q}(f_T(N_Q, N_Q).T + k.T^{con}) \tag{3.9}$$
$$c_P^{block.A} = f_P(k.N_Q, N_Q).P \tag{3.10}$$

where the T^{con} is the computational cost of the sub-block concatenation operation.

B - the concatenation is no beneficial or possible, so the processing is executed on each sub-block and the sub-results are concatenated to form final result:

$$c_T^{block.B} = \frac{1}{N_Q} f_T(N_Q, N_Q).T \tag{3.11}$$
$$c_P^{block.B} = (k.f_P(N_Q, N_Q).P + P^{con}) \tag{3.12}$$

where the P^{con} is the computational cost of the sub-result concatenation operation.

In comparison with previous schemas, in *simple flow*, each reference sample was used N_Q times for training, in *large-step flow*, k times and in *block-based flow* just ones.

The *flowing window concept* is discussed for its utilization for visual codebooks techniques but the concept is applicable to other approaches as well.

3.2 Dynamic Inverse Document Frequency

The Inverse Document Frequency weights computed by weighting function *idf* (Eq. 2.4) emphasizes the rare visual words and down-weights the frequent ones. In standard static approach, the visual word weights are computed during the codebook training and keep the same when codebook is used to translate features from data of any application domain. When the codebook is trained on big enough dataset containing data from many application domains, variability of the weights of visual words decreases as each application domain introduces data with different image contents. Having the codebook trained on data from one application domain and using it on data from different application domain, the codebook performance should be improved by recomputing the visual word weights on data from particular application domain.

The same idea can serve to improve the codebook performance working with on-line data. Not only that is highly probable that each video will contain data from different application domain (news, documentary, sports, movies, music clips, cartoons, etc.), but also one video can contain sub-sequences of different types. The adapted IDF weights can reflect the diversity and changes in data types.

Based on *flowing window concept*, the dynamic IDF of the i^{th} visual word is computed as logarithm of ratio of number of documents containing the word $|\mathcal{D}(w_i)|$ and number of subset of the documents $\mathcal{D}_\mathcal{B}$ (Eq. 3.13).

$$idf_i^{dyn} = \log \frac{|\mathcal{D}_\mathcal{B}|}{|\mathcal{D}_\mathcal{B}(w_i)|} \tag{3.13}$$

Standard construction of the image signature computes the IDF weights directly. When using dynamic IDF, the standard concept must be slightly changed. During construction of the image signature, only term-frequencies tf are used. The IDF weights is then used in next operations separately. The reason for separating the IDF weights from the image signature is, that the dynamic IDF can be updated in each frame. When the IDF weights would be encoded in the image signatures, re-weighting of the visual words in the reference block would be expensive.

The dynamic IDF can be adapted on-the-fly, adding the words occurrences with an incoming new frame and removing the occurrences of old block-leaving frame. Due to the fact that the realization of the distribution computation introduce add and remove operations adapting the dynamic IDF just by one single sample, the cost functions can be defined as follow.

$$f_T(N_T, N_P) = \begin{cases} 2N_P, & \text{if } N_P < \frac{N_T}{2} \\ N_T, & \text{otherwise} \end{cases} \tag{3.14}$$

$$f_P(N_T, N_P) = 1 \tag{3.15}$$

The evaluating of the best flow realization schema can be based on following analysis.

$$c^{simple} = 2T + P \tag{3.16}$$

$$c^{large} = \begin{cases} 2T + P, & \text{if } N_Q < \frac{N_R+N_Q}{2} \\ \frac{N_R+N_Q}{N_Q}T + P, & \text{otherwise} \end{cases} \tag{3.17}$$

$$c^{block.A} = \frac{k.T^{con}}{N_Q} + T + P$$

$$\approx \frac{k+1}{N_Q}T + P; \text{ if } T \cong T^{con} \tag{3.18}$$

$$c^{block.B} = T + k.P + P^{con} \tag{3.19}$$

For the usual configuration when $N_Q < N_R$, the *simple flow* (Eq. 3.16) and *large flow* (Eq. 3.17) have the same computational cost. Besides the extremum case when $N_Q = N_R = 1$, the *block-based flow (A)* (Eq. 3.18) has the lowest computational cost, as the expression $\frac{k+1}{N_Q}$ is < 1.

The *block-based flow (A)* is the beneficial realization for approaches that introduce add and remove updating operations and where the error introduced by the increased amount of training samples is insignificant.

3.3 Flowing Inverted File Index

Some of the computer vision applications deals with image retrieval. In the video processing applications, such tasks can be for example detection of self-similar video parts, frame searching or searching for the frame position it the video sequence. Image retrieval solutions based on bag-of-words usually use the inverted file index [15].

An inverted index stores word occurrences which maps individual words to the documents in which they occur. From the two variant of inverted indexes, one called *full inverted index* contains also the positions of the words in a document and second one called *inverted file index* stores only references to the documents [1], the computer vision approaches utilize the latter.

For the dynamically changing content, the adaptable version of the inverted file index is proposed to fit in the *flowing window concept*. The index is a list of visual words addressed by their identifier. The visual word identifiers are usually the integer values, so they can be used to address the list of documents itself without any additional hashing structure. An example structure of the inverted file index is on Fig. 3.2.

For each visual word, the index contains the ordered list of images where the visual word occurs. The images are sorted by the time they appeared in the video. Such structure can be quickly updated; to add or remove the image signature to, resp. from the index, the image is added, resp. removed from the list for each visual word in image signature. Keeping the list of images for each visual word ordered significantly increase the speed of the updating operation.

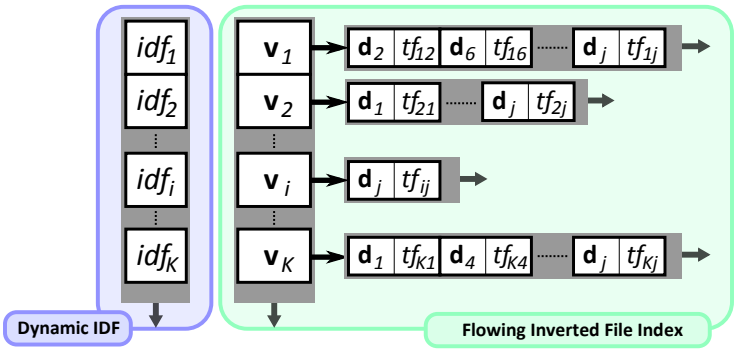


Figure 3.2: Flowing inverted file index structure.

As the adapting operations for the flowing inverted file index are realized in constant time, so the analysis for the best realization of the *flowing window concept* would result similar to the dynamic inverse document frequency (Eq. 3.16–3.19).

3.4 Adaptable Visual Codebook

The *adaptable visual codebook* is designed to reflect the characteristics of the present data but the changes in the codebook structure must keep the previous *image signatures* valid.

The first requirement is fulfilled by adding a new visual words into the vocabulary, when no similar visual word exists in the codebook. The similarity threshold is the main parameter of the method. This parameter is called the *word-size* parameter as it defines the occupied space around the visual word in the descriptor space. The visual word positions in the descriptor space are stable, the adaptability is realized only by adding the new words into the codebook. No changes of the visual word positions are allowed. The stability of the visual words fulfill the second demand to keep the previous translations valid.

The adaptable visual codebook is realized by the kd-tree structure. Avoiding the construction of the kd-tree for each new visual word, the temporary buffer is used to manage the new visual words before there are incorporated into the kd-tree itself (see Fig. 3.3). Using the *flowing window concept*, the total cost functions are defined for the adaptation and the search steps for the large-step flow and the block-based flow approach. As the simple flow approach is a limitary case of the large-step flow approach, the simple flow is not discussed in detail.

The method does not introduce the operations for samples removal from the structure (only invalidation of particular visual words can be realized). Using the *simple flow* approach would cause the unlimitedly increasing size of the visual codebook so the visual codebook cannot be trained on-the-fly and need to be constructed before each processing. Due to this fact, the *large-step flow* approach is always less expensive than the *simple flow* so the *simple flow* approach analysis is omitted.

The cost function of the kd-tree construction is $N \log_2(N)$ and adding to the temporary buffer is constant. During the processing step, the used cost functions

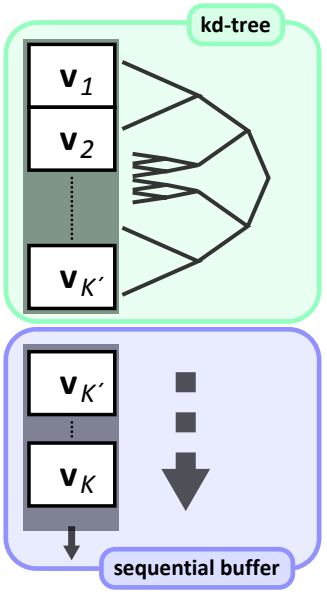


Figure 3.3: Structure of the adaptable visual codebook.

are $log_2(N)$ for kd-tree search and $\frac{N}{2}$ for buffer search. For the clarity reason, the costs of elementary operations will be in the analytical analysis omitted.

For the *large-step flow* approach, the total cost c^{large} is defined in Eq. 3.20 as the kd-tree construction over N_R samples and sequential buffer adaptation for N_Q samples. The computational cost of the searching in the buffer reflects its increasing size.

$$\begin{aligned} c_T^{large} &= \frac{1}{N_Q}(N_R \log_2(N_R) + N_Q) \\ c_P^{large} &= \log_2(N_R) + \frac{N_Q}{2} \\ c^{large} &= \frac{N_R + N_Q}{N_Q} \log_2(N_R) + \frac{N_Q}{2} + 1 \end{aligned} \tag{3.20}$$

The location of the extremum is determined by taking the derivative of the function c^{large} with respect to N_Q and setting it to zero.

$$\begin{aligned} \frac{\partial c_{large}}{\partial N_Q} &= 0 \\ \frac{N_Q^2 - 2N_R \log_2(N_R)}{2N_Q^2} &= 0 \\ N_Q &= \pm\sqrt{2N_R}\sqrt{\log_2(N_R)} \end{aligned} \tag{3.21}$$

Only positive case of the solution for N_Q in Eq. 3.21 is usable expressing the relation between N_R and N_Q for the best *large-step flow* configuration.

For the *block-based flow (B)* approach, the total cost $c^{block.B}$ is defined in Eq. 3.22, where kd-tree is constructed for each sub-block with the N_Q samples. The searching

is executed in $(k-1)$ sub-blocks and one temporary buffer.

$$\begin{aligned} c_T^{block.B} &= \frac{1}{N_Q}(N_Q \log_2(N_Q) + N_Q) \\ c_P^{block.B} &= (k-1)\log_2(N_Q) + \frac{N_Q}{2} \\ c^{block.B} &= \frac{N_R + N_Q}{N_Q}\log_2(N_Q) + \frac{1}{2} + 1 \end{aligned} \tag{3.22}$$

The extremum of the function $c^{block.B}$ is again determined by taking the derivative of the function with respect to N_Q and setting it to zero.

$$\begin{aligned} \frac{\partial c^{block.B}}{\partial N_Q} &= 0 \\ \frac{1}{N_Q^2}(N_Q + N_R - N_R \log_2(N_Q)) &= 0 \end{aligned} \tag{3.23}$$

This model example suppose that the particular training costs are more or less similar to processing costs. In this case, the solution in Eq. 3.23 has no applicable solution as the N_Q is noticeably higher than N_R in all cases. When the costs of the training and of the processing stages are going to be considerably different, the in Eq. 3.23 might change.

The *block-based flow (A)* approach is not analyzed in detail as the merging of the tree structures in the sub-blocks would be hardly realizable in acceptable cost.

Chapter 4

Experiments

The experiments with the novel adaptable method and the selected state-of-the-art techniques are evaluated on the image retrieval task.

4.1 Data Collections

The selection of utilized data for experiments reflects the needs of the two types of experiments - the image collections with manual annotations for tasks based on image retrieval and video collections for on-line processing where no content annotation is needed.

Kentucky Image Collections. The Kentucky image collection is designed for image or object retrieval evaluation tasks. The dataset is provided with the manual annotation, so the nature or a class of the image is known. The *Kentucky* dataset was created by Nister and Stewenius [12] as a recognition benchmark at Kentucky University. The set consists of 2550 groups of 4 images each, that is 10200 images in total. The objects images are taken from different angles and rotations. The size of the images is approximately 640x480 pixels. The dataset contains also images of CD covers, that are rather easier visual concepts to then for example natural objects as flowers. The images also differ in sharpness; many of the indoor images are somewhat blurry. The examples of image samples is are on Fig. 4.1.

Figure 4.1: Examples from Kentucky image collection. [12]

TRECVID Video Collections. The TRECVID is on-going series of workshops focusing on a list of different information retrieval (IR) research areas in content based retrieval of video. The *TRECVID* videos are coded according to MPEG1 with 25 fps and bitrate 1152Kbps and the frame size is 352x288 pixels. Most of the videos from the used collections have durations approximately 30 minutes. The example frames from the video collection are in Fig. 4.2. As the *tv.bbc* data is not annotated, no key-frames are defined.

Table 4.1: The number of images and features in TRECVID video datasets.

collection name	hours	files	key-frames	used-frames
tv7.sv.devel	50	110	18.012	4.791.166
tv7.sv.test	50	109	18.142	4.572.925
tv8.sv.test	100	219	35.766	9.360.806
tv9.sv.test	180	400	61.384	16.708.689
tv.bbc.devel	35	91	-	137.169

The *tv.sv* dataset is the union of all *tv*.sv* datasets. It is the collection of videos used over the years in TRECVID. The collection contains news magazine, science news, news reports, documentaries or educational programming. The overview of hours of video divided by years is given in Tab. 4.1.

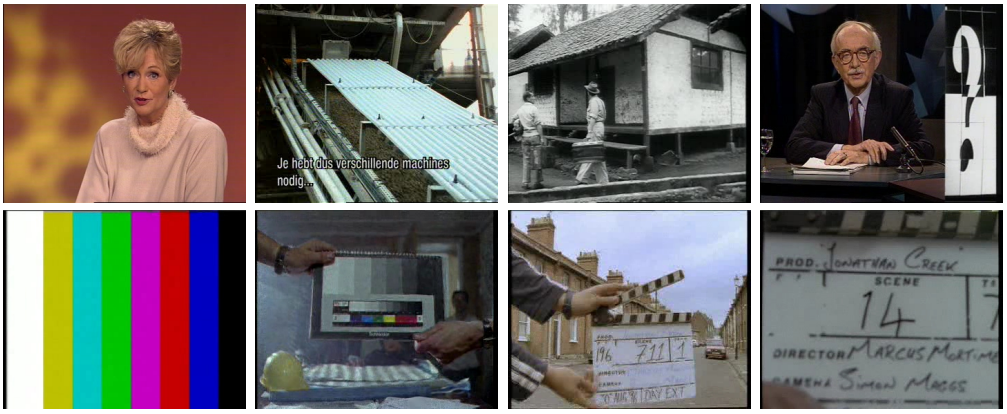


Figure 4.2: Example frames from TRECVID dataset - tv.sv (top row) and tv.bbc (bottom row). [17]

The *tv.bbc* dataset is a collection of videos provided by The BBC Archive containing unedited video material - rushes. Rushes are the raw material used to produce a final video. 20 to 40 times as much material may be shot as actually becomes part of the finished product. Rushes contain many frames or sequences of frames that are highly repetitive, e.g., many takes of the same scene redone due to errors, long segments in which the camera is fixed on a given scene or barely moving, etc. A significant part of the material might qualify as stock footage - reusable shots of people, objects, events, locations, etc. Many of the videos begin with initializa-

tion shot (cca 40-60 seconds) with color initialization stripes. Video sequences also contain junk frames - frames with color stripes, clapperboards, etc.

Transformed Collections. The evaluation of the experiments dealing with video processing are mostly based on comparison of the approach performances on the original and transformed data. The transformed video data are created artificially from the originals. In the experiments presented in this work, the following video transformations and distortions are addressed: uniform scale transformation, partial occlusion of the visual content by static banner, blur and white noise. The experiments based on the video processing uses each 25_{th} frame in the video sequences, i.e. the length of the reference block 600 refers to the 10 minutes of video content. The further details and configurations about used transformation are described in particular experiments.

The examples of the simulated transformations and distortions in Fig. 4.3 show the overlapping by the banner covering 30% of the image height (top-left), uniform scale change to 90% (top-right), added 25% of the white noise (bottom-left) and blur by Gaussian smoothing with $\sigma = 3.0$ (bottom-right).

Figure 4.3: Examples of video transformations and distortions.

4.2 Evaluation metrics

Mean Average Precision. Most of the evaluations of vocabulary characteristics were based on image retrieval performance. The image retrieval system evaluation is often based on the *mean Average Precision* computation [3, 13, 19]. The result of the query of the image retrieval system is a ranked list of images. It is desirable to consider the order in which the returned images are presented. Average Precision (AP) represents the area under Precision-Recall curve [15] for a query. The AP can be also computed from the ranked list of retrieved images [19] (Eq. 4.1).

$$AP = \frac{1}{m} \sum_{i=1}^{n} \frac{relevant(x_i)}{i} \tag{4.1}$$

The n is the number of retrieved images, m is the number of relevant images, x_i is the i-th image in the ranked list of retrieved images $X = \{x_1, \ldots, x_n\}$ and $relevant(x_i)$ returns the number of relevant images in the first i images, only if the x_i is relevant image itself, and 0 otherwise. This measure gives a number in range (0, 1] where a higher number corresponds to a better performance.

Image Retrieval Ratio. Experiments based on unknown (unannotated) video data are facing the problem, when it is quite difficult to decide whether two video frames are *similar* or *not*. Even when the data is annotated, the similar video content does not really mean that the video frames are similar. Video similarity search task is often performed on slightly distorted videos.

To solve this problem, this work introduces a metric that is based on selection of the smallest subset of frames that contains the query frame with a given probability. The *Image Retrieval Ratio* (IRR) is expressed as the ratio of the number of frames in the smallest subset to the length of the reference block.

It is assumed that the results of a search in the series of frames in applications is done so that the maximum value of the similarity metric is taken. The maximum value is multiplied by a constant $c < 1$ to get a threshold $t(c)$ (Eq. 4.2). All of the frames with similarity above such threshold are considered for further processing as they may be *similar enough* to the query image. The constant c, to multiply the maximum value with, is calculated based on statistics of video sequences so that with some pre-defined probability p, the *ground truth* frames are contained in the set of selected ones.

Let $\mathcal{D}_\mathcal{R}$ be the set of frames in reference data block (see Eq. 3.2). The probability p of occurrence of the *ground truth* frames in the selected set $\mathcal{S}$ is defined in Eq. 4.5.

$$t(c) = c * \max_{f \in \mathcal{D}_\mathcal{R}}(sim^{cos}(\mathbf{q}, \mathbf{f})) \tag{4.2}$$

$$\mathcal{S}(\mathbf{q}, \mathcal{D}_\mathcal{R}, c) = \{\mathbf{f} \in \mathcal{D}_\mathcal{R} | sim^{cos}(\mathbf{q}, \mathbf{f}) \geq t(c)\} \tag{4.3}$$

$$p = (\mathbf{q} \in \mathcal{S}(\mathbf{q}, \mathcal{D}_\mathcal{R}, c)) \tag{4.4}$$

$$m = |\mathcal{S}(\mathbf{q}, \mathcal{D}_\mathcal{R}, c)| \tag{4.5}$$

where $\mathbf{q}$ is the searched frame, $\mathcal{S} \subseteq \mathcal{D}_\mathcal{R}$ and c is some constant. Then the size of the set $\mathcal{S}$ can be seen as the metric of success m - the smaller the size of $\mathcal{S}$, the better.

In presented experiments, the value of c is computed as percentile of the best similarity scores over the all queries in the experiment run. The evaluation procedure of the most of the experiments working with video collection is following:

1. randomly take one video from the video collection,
2. randomly select the reference block of the given size N_R,
3. randomly select the query block of the given size N_Q within the range of the reference block from the transformed video sample,
4. execute particular methods on the reference and query blocks,
5. pick-up in random order samples form the query block and search the reference block,
6. track all similarity scores between the query frame and all frame from the reference block.

Given the similarity score tracks from the procedure together with the ground truth frame for each track, the m can be computed for several values of c. The result of such one experimental sub-run may look like in Fig. 4.4.

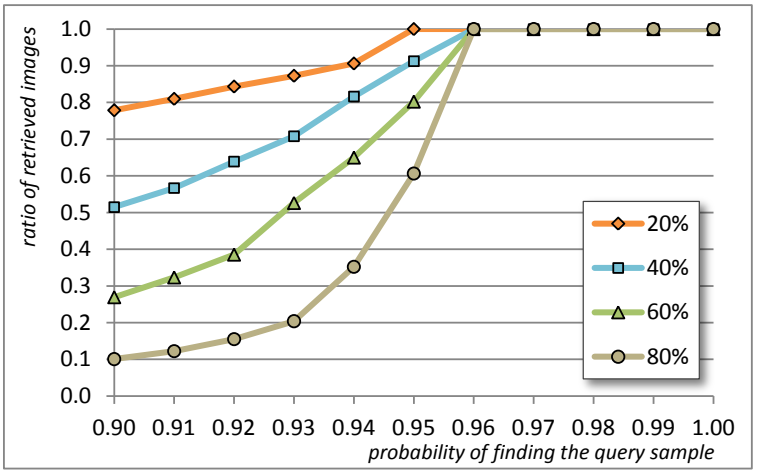


Figure 4.4: Example sub-result when computing Image Retrieval Ratio value.

The c parameter corresponds to the probability that retrieved frames contains the query sample. The vertical axis then for range of probability p values shows, how many frames must be retrieved out of the reference data to fulfill the retrieval requirement. The number of retrieved frames is normalized by the size of the reference block. The smaller the number of retrieved frames for given probability the better performance of the method.

The procedure is executed several times with different level of distortion; in Fig. 4.4 the resize to 20%, 40%, 60% and 80% is used. The results show how the number of retrieved frames grows with increasing parameter of the distortion. The IRR measure is than taken as the average of the normalized areas under the each particular curve. The measurement value for given example is then $IRR = 0.7693$.

4.3 Image Local Features

Although the SIFT and SURF feature extraction and description methods have been already evaluated before, e.g. [2, 11], following experiments aims to compare their performance and computational cost with two non-standard approaches. Following experiments are focused on evaluation of the image local feature extractors in the image retrieval pipeline based on visual codebooks. The experiments are based on *Kentucky dataset*.

The experiments evaluating the performance of the feature extractors are executed using various codebook settings. Fig. 4.5 (left) shows two results of the precision measurements out of all configurations - *normal* and the *best*. The *best* and the *normal* results are achieved using codebook configurations shown in Tab. 4.2 (the σ factor and *knn* parameter are discussed in more detail in following section). The performance is measured and represented using mAP (Eq. 4.1) value.

The overall comparison (Fig. 4.5 (right)) shows the price of the higher retrieval performance of the *best* configuration to the *normal* configuration. The **precision** shows, how much is the *best* solution better than the *normal* one and the **cost** shows

Table 4.2: Visual codebook configurations used for image feature extractors evaluation.

	normal	best			
params	all	SIFT	SURF	FHOG	FOGH
cb. size	10^4	10^6	10^6	10^3	10^5
σ factor	1.0	1.0	1.0	1.0	1.0
$k$nn	5	5	5	20	1

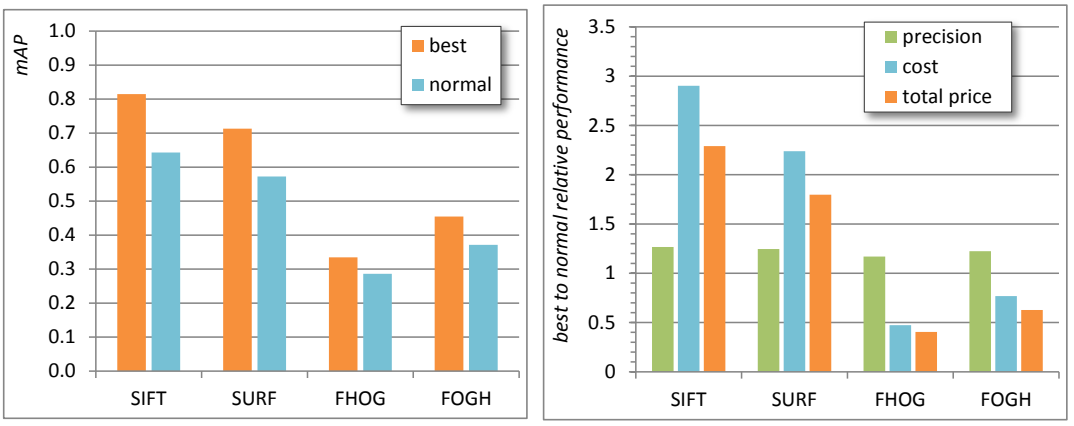


Figure 4.5: Comparison of the retrieval performance of the feature extraction methods using their *normal* and *best* codebook configuration (left) and the cost of the improvement (right).

how much expensive is this precision improvement. The **total price** represents the ratio of the performance and the price, so the value higher than 1 means, that the precision performance is too expensive and value below 1 means the cheaper performance. As the focus is put on the methods' computational costs, SIFT and SURF methods are not worth to use in the *best* configuration.

Based on experiments on the Kentucky dataset, the average numbers of extracted features per one image for each feature type is shown together with the extraction time in Tab. 4.3. Note, that the computational costs are directly dependent on the amount of the extracted features. The times are measured and averaged on the several machines with different configuration (Intel/AMD, cca 2.4GHz, up to 1GB of RAM, using just one core in multicore processors).

The normal configuration is used as the basic configuration for most of the experiments with the introduced adaptable method.

Table 4.3: Average number of extracted features per image and extraction times.

	SIFT	SURF	FHOG	FOGH
features	1290	641	581	1156
time[s]	1.62	0.23	0.05	0.37

4.4 Codebook attributes

Searching Strategy. The following experiments aims to evaluate the clustering performance degradation when approximated search is used instead of the sequential one. The results of the experiments also study the real performance improvement. The clustering performance is measured using the clustering error. Both results, the error and computational cost, are represented relatively to sequential approach. The results are evaluated on the four sizes of visual codebooks (10^2, 10^3, 10^4, and 10^5). The training of the bigger visual codebooks with sequential search is too computational expensive.

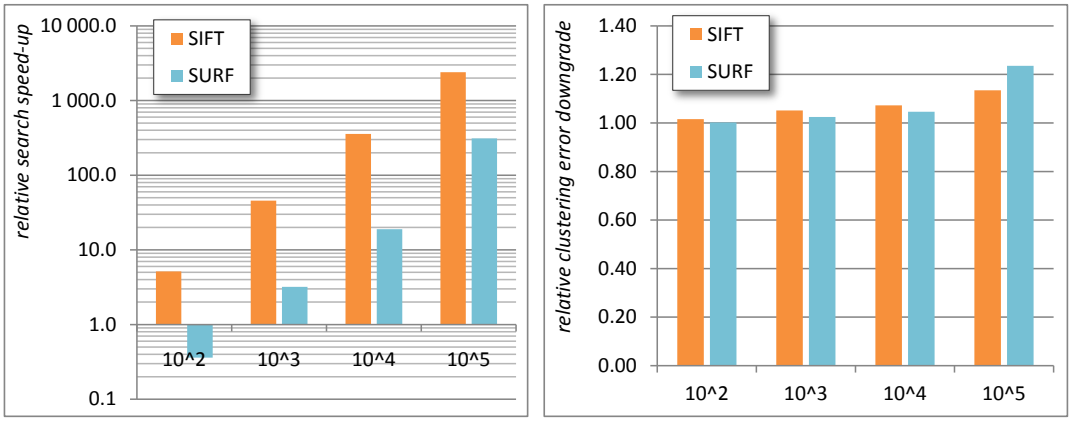


Figure 4.6: Precision and speed performance of the visual codebooks with different sizes and projection strategies.

The values in Fig. 4.6 represent the ratios of the time consumptions and the clustering errors between the sequential approach and the kd-tree based structure. The results show that even if the sequential search strategy outperforms the approximated nearest neighbor search that is realized by the kd-tree structure, the computational cost is exponentially higher in comparison with slightly better performance. The further experiments utilizes the kd-tree architecture in the visual codebook realizations for its high improvement in the method's speed and low precision degradation.

Soft-Assignment Weighting. The soft-assignment techniques theoretically improves the approximation error that is introduced by descriptor space quantization. The experiment evaluating the performance improvement when increasing the amount of nearest neighbors k are made for knn 1, 5, and 20. For each particular image feature type, the performance is evaluated for all codebook sizes (10^2, 10^3, 10^4, 10^5, and 10^6) with exponential weighting function (Eq. 2.5). The experiments, where $knn = 1$, represents the hard-assignment approach.

The results in Fig. 4.7 show the image retrieval performance improvement, when using soft-assignment approach. The improvement for SIFT and SURF features is in several palaces almost 20%. The performance of the FHOG and FOGH features is negatively influenced by the soft-assignment. As the utilization of the soft-assignment approach costs some computational power, the latter image features are

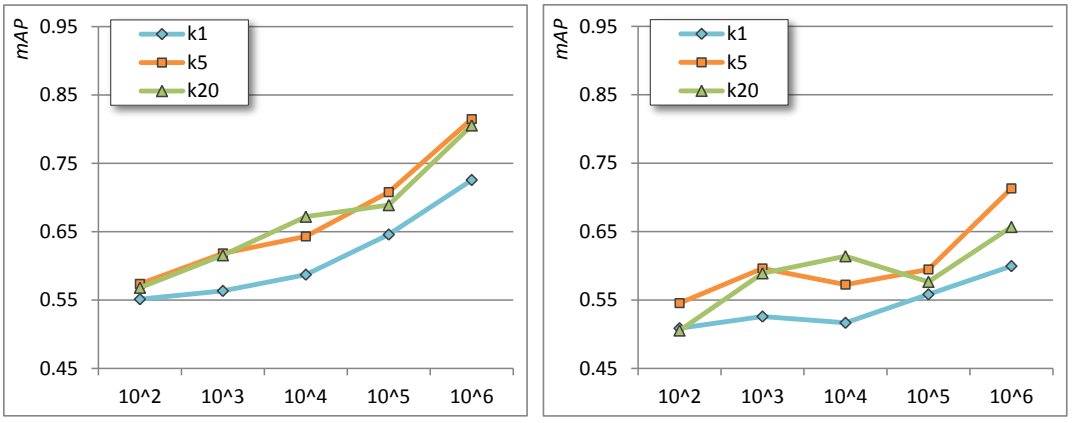


Figure 4.7: Soft-assignment functions for various feature types - SIFT (left), SURF (right), and for different codebook sizes.

better translating using hard-assignment.

4.5 Dynamic Inverse Document Frequency

The experimental analysis of the dynamic inverse document frequency procedure is focused on the improvement of the video searching precision when the adaptable method is introduced. The dynamic adaptability is based on the *flowing window concept*. The experiments are evaluated using video searching task using the Image Retrieval Ratio (Eq. 4.5) that measures the retrieval performance on unannotated video data. Note, that the lower IRR value the better, as the smaller amount of reference frames are retrieved with the given probability that the retrieve data contains the query sample.

The experiments with different IDF types (static, dynamic) aim to prove or reject the assumption, that computing the IDF weights on actual data instead of using weights from training on different data improve the image retrieval performance. The results are represented using the ratio between IRR computed using dynamically updated weights and IRR computed with static weights - *IRR ratio*. The static weights are from the vocabulary training stage and the dynamic weights are computed for the data from the reference and the query blocks. The smaller the ratio is the better, as the IRR represents the amount of retrieved data which should be as small as possible.

The first experiment analyzes the soft-assignment approach utilized together with dynamic IDF. The exponential weighting function (Eq. 2.5) has a parameter σ that influence the significance of k nearest neighbors by their distance to the single descriptor. To study the influence of the σ, three values of σ are tested: *small*, *normal* and *large*, where the *small* means 10% of the *normal* σ value and the *large* means 1000%. The *normal* σ differs for each particular feature type. The values were experimentally set to $\sigma^2 = 0.1$ for SURF, FHOG and FOGH descriptors and $\sigma^2 = 6125$ for the SIFT descriptor.

The SURF feature type is used with the codebook size 10^4. The length of the

reference block is $N_R = 600$. The experiments run for different configurations varying in *knn* (1, 5, 20) and σ factor (*small*, *normal*, *large*). The results in Fig. 4.8 (left) approve the assumption that the improvement of the dynamically adapted IDF increases when more visual world are taken into account during translation.

The next experiments aims to evaluate the influence of the codebook size to the dynamic IDF performance. Three codebook sizes are tested - 10^3, 10^4, and 10^5, using the SURF features and $knn = 5$ and *normal* σ factor in soft-assignment. The results in Fig. 4.8 (right) shows the decreasing improvement of the dynamic IDF approach over the static one, as the codebook size increases. It may be caused by the more regular distribution of the used visual words in the reference data. It decreases the variability of the IDF weights computed from this data block and makes the static and dynamic IDF weights similar.

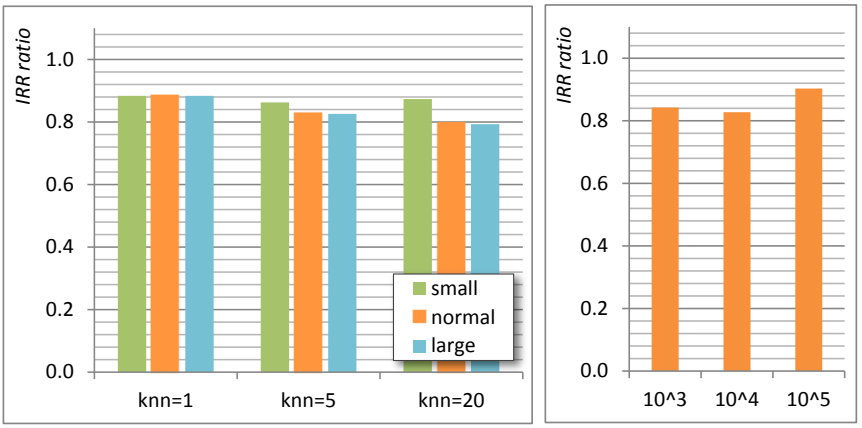


Figure 4.8: Dynamic IDF tested various soft-assignment configurations (left) and different codebook sizes (right).

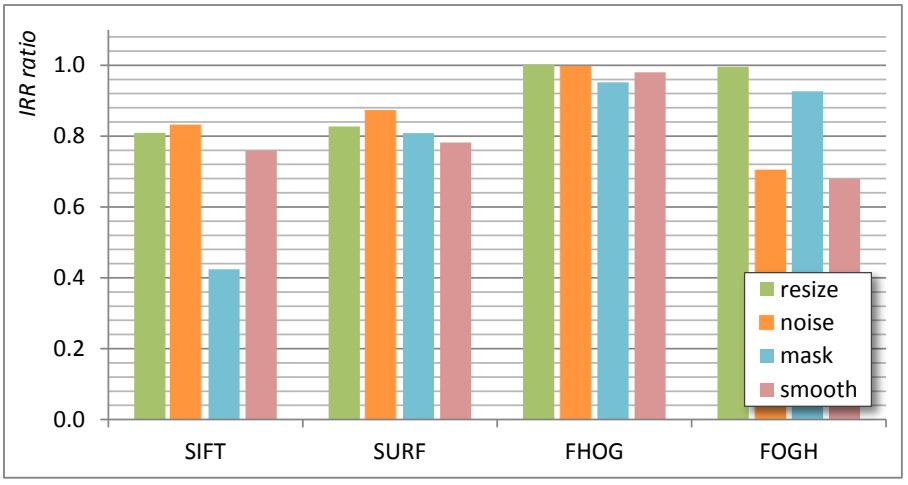


Figure 4.9: Dynamic IDF tested under various distortion transformations.

The performance of the feature extraction methods for the video searching task

is evaluated on the four types of the video distortion transformations. The results in Fig. 4.9 are computed using the following transformation settings: resize (80%, 60%, 40%, and 20%), noise (10%, 20%, 30%, and 40%), mask (occluding 10%, 20%, 30%, 40%, and 50% of the frame), and smooth (using Gaussian blur with σ=1,2,4 and 8). The experiments utilized the codebook size 10^4 and with $knn = 5$ and *normal* σ factor in soft-assignment.

The results of the experiments approved the assumption, that the introduced adaptable method has lower overhead of the retrieved samples when using in video searching task. The usage of the dynamic IDF for SIFT and SURF feature types decreases the amount of retrieved data by 20% in average. The improvement of the dynamic approach is not notably beneficial for FHOG and FOGH feature types.

4.6 Flowing Inverted File Index

Experiments analyzing the performance of the flowing inverted file index are focused on the improvement of the computational cost efficiency. The results in Fig. 4.10 (left) show the speed-up when the flowing inverted file index is used instead of the sequential search. The compared times also include the time of the index construction and updates. The flowing index is tested with various soft-assignment settings (knn=1, 5, and 20 and various σ factor) and with the codebook size 10^4. The influence of the different codebook sizes to the index performance is shown in Fig. 4.10 (right) for $knn = 5$ and *normal* σ factor. Both experiments use the SURF features.

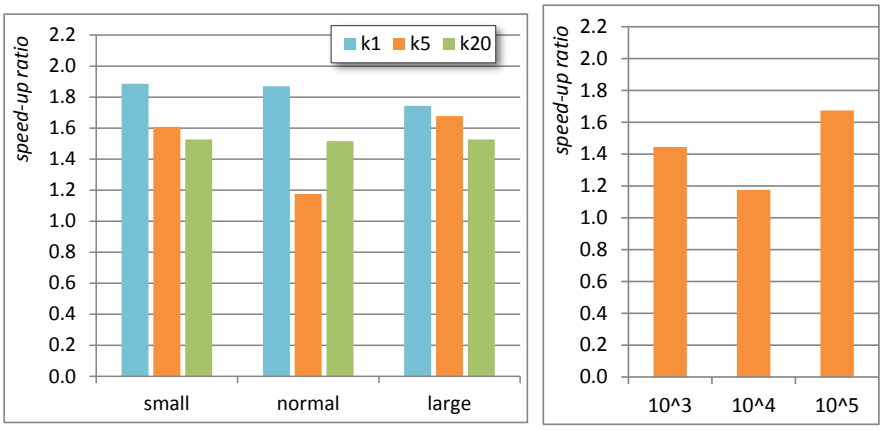


Figure 4.10: Flowing Inverted File Index

4.7 Adaptable Visual Codebook

The main fundamental parameter of the adaptable method for the visual codebook construction is the size of the visual word in the descriptor space - the *visual word size* or VWS. Again, similar to σ factor in soft-assignment weighting function, the VWS vary for different feature descriptor types.

The VWS values are experimentally chosen from the measured distributions of distances between the descriptors and theirs k nearest visual words. The distributions are measured using *Kentucky* dataset with $knn = 10$. Based on the measured distributions, the three *visual word sizes* are selected for each feature type. As the values differ for each feature type, the values are labeled as *small*, *normal* and *large* sizes representing the VWS values as shown in Tab. 4.4.

Table 4.4: Selected visual word sizes with assigned labels.

VWS type	SIFT	SURF	FHOG	FOGH
small	300	0.5	0.5	0.8
normal	350	0.6	0.6	0.9
large	400	0.7	0.7	1.0

The next experiments evaluate the influence of the *visual word size* to the precision of the image retrieval (based on IRR measurement) with various soft-assignment configuration and different feature types. The results of the different knn values with *normal* σ factor and using SURF feature are shown in Fig. 4.11 (left) and the results for the different feature types are on the right. The results are represented using the ratio between IRR computed using adaptable method and IRR computed with static visual codebook - *IRR ratio*. Both results in Fig. 4.11 show, that the smaller *visual word size* is the better performance (the lower IRR ratio is achieved).

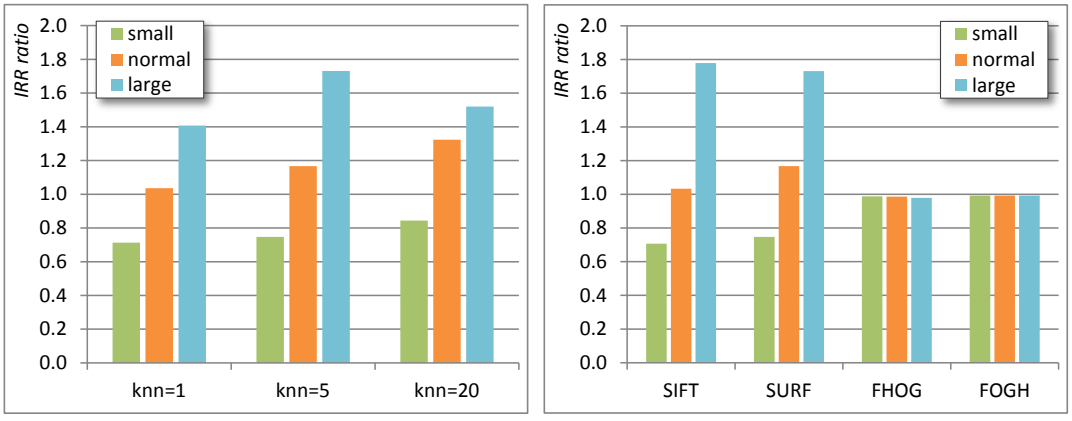


Figure 4.11: Results of various soft-assignment settings (left) and different feature types (right) with different visual word sizes.

The experiments with the smaller *visual word sizes* than the *small* ones result in visual codebooks with too many visual words usually representing just one single feature descriptor. Such visual codebook has very low ability to generalize the similar feature descriptors so the precision of such visual codebook used in retrieval tasks rapidly decreases. The further experiments are based on the adaptable visual codebooks with the *small VWS*.

The influence of the size of the reference block is evaluated with the following three N_R lengths: 600, 1200, and 1800. As the videos are sampled with 25fps (frames

per second) and each 25^{th} frame is processed in experiments, the N_R sizes process 10, 20, and 30 minutes long video parts. The results in Fig. 4.12 (left) show the slightly increasing improvement when more data is used during the training stage.

Influence of the data domain to the experimental results is avoided by evaluating the adaptable method on different dataset. The *tv.sv* dataset is used and the results in Fig. 4.12 (right) again prove the assumption that the performance of the retrieval method based on the visual codebook is improved when the method is able to adapt to the actually processed data.

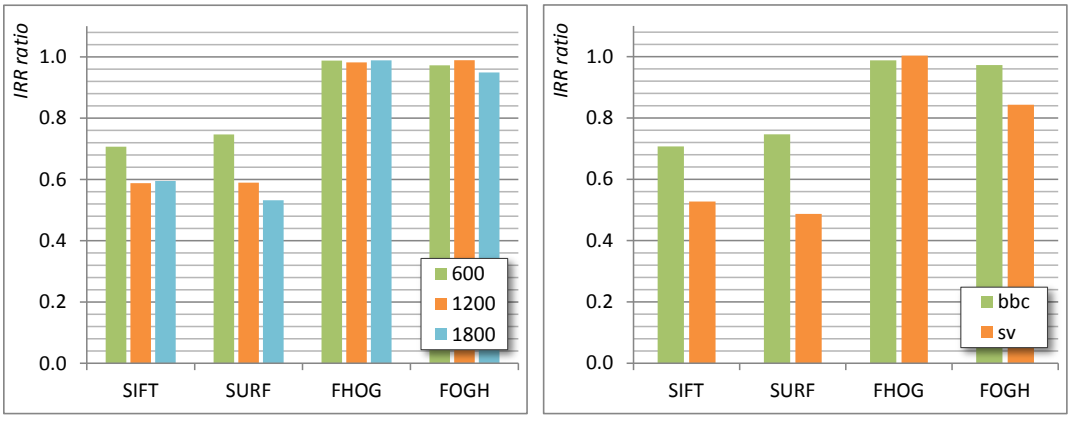


Figure 4.12: Adaptable method performance with various length of the reference block (left) and improvement tested on different datasets (right).

The performance of the adaptable visual codebook for different feature types in the video searching task is evaluated again on four types of the video distortion transformations. The results in Fig. 4.13 are computed using the following transformation settings: resize (80%, 60%, and 40%), noise (10%, 25%, and 40%), mask (occluding 10%, 30%, and 50% of the frame), and smooth (realized by Gaussian blur with σ =1,3, and 9). The experiments utilizes the codebook size 10^4 and with $knn = 5$ and *normal* σ factor in soft-assignment. The static codebook is trained on the *tv.bbc* dataset.

The results in Fig. 4.13 show the precision of the adaptable visual codebook under various distortion transformations. The performance of the method is represented using the IRR measurement (top) and the IRR ratio (bottom). All of the tested feature types outperforms the static approach when applied in adaptable method. This is true for all tested image distortion transformations. However, the improvement of FHOG and FOGH is disputable when the precision is very low for both approaches.

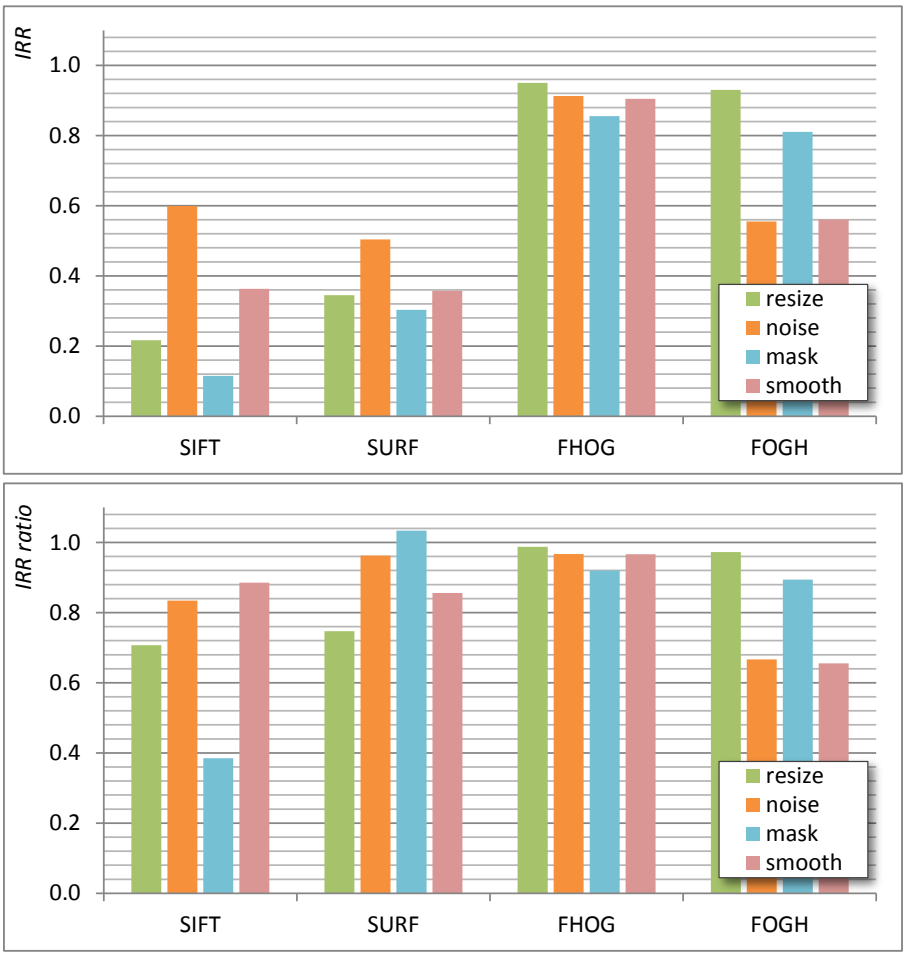


Figure 4.13: Adaptable visual codebook performance tested under various distortion transformations (top) and the relative improvement over the static approach (bottom).

Chapter 5

Conclusions

In this work, I introduce the new adaptable method for on-line video searching in real-time based on visual codebook. The new method addresses the high computational efficiency and retrieval performance when used on on-line data.

My novel adaptable method is based on the newly introduced *flowing window concept* that defines the ways of selection of data, both for system adaptation and for processing itself. Together with the concept, the computational cost functions are defined to find the best configuration when applying the concept to some new method. The analytical analysis of the proposed adaptable procedures demonstrate the usage of this mathematical background.

The adaptable method originates in procedures utilized by static visual codebook techniques. These standard procedures are modified to be able to adapt to changing data. The procedures that improve the new method adaptability are: dynamic inverse document frequency, adaptable visual codebook and flowing inverted index. The developed adaptable method was experimentally evaluated and the presented results show how the adaptable method outperforms the static approaches when evaluating on the video searching tasks.

During the realization of the goal of this work, the framework for evaluation of video processing methods was designed and implemented. The evaluation is based on the Image Retrieval Ratio metric. The video collection used in the experiments originates form the TRECVID workshop . The method was evaluated on video collection with various challenging image distortions and conditions. The preliminary results of the introduced method were successfully used in recent years in the TRECVID evaluations.

The work studies the selected state-of-the-art methods for image description and visual codebook construction and their potentials for the real-time approach. The experimental analysis of the selected methods are realized on the designed and implemented image retrieval framework. The evaluation is based on the mean Average Precision metric often used in image retrieval systems. The experiment were executed and evaluated on annotated image collection from Kentucky university.

My adaptable method introduces general approaches that can be used also for other video processing tasks than video searching. Applying the results of this work also on other video processing tasks is one the goal of future work. Also some new promising image feature extraction methods may be experimentally evaluated with

the *flowing window concept*.

The practical application of the adaptable method is particularly in the video processing systems where significant changes of the data domain, unknown in advance, is expected. The method is applicable in embedded systems monitoring and analyzing the broadcasted TV on-line signals in real-time.

Bibliography

[1] Baeza-Yates, R. A., and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

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

[3] Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil* (2007).

[4] Dalal, N., Triggs, B., and Schmid, C. Human detection using oriented histograms of flow and appearance. In *In European Conference on Computer Vision* (2006), Springer.

[5] Jiang, Y.-G., Ngo, C.-W., and Yang, J. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval* (New York, NY, USA, 2007), ACM, pp. 494–501.

[6] 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.

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

[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. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell. 27*, 10 (2005), 1615–1630.

[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] Nister, D., and Stewenius, H. Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 2161–2168.

[13] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[14] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[15] Richard O. Duda, P. E. H., and Stork, D. G. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

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

[17] Smeaton, A. F., Over, P., and Kraaij, W. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM international workshop on Multimedia information retrieval* (New York, NY, USA, 2006), ACM, pp. 321–330.

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

[19] Uijlings, J. R. R., Smeulders, A. W. M., and Scha, R. J. H. Real-time bag of words, approximately. In *CIVR '09: Proceeding of the ACM International Conference on Image and Video Retrieval* (New York, NY, USA, 2009), ACM, pp. 1–8.

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